AWS Elemental Live: User Guide
Copyright © 2020 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.
# Table of Contents

What Is AWS Elemental Live? ............................................................................................................. v
Finding Information About AWS Elemental Live ............................................................................. 1
Interfaces for AWS Elemental Live ..................................................................................................... 1
How AWS Elemental Live Works ........................................................................................................ 2
Terminology ........................................................................................................................................ 2

Inputs ................................................................................................................................................ 4
Interleave 2SI Inputs .......................................................................................................................... 4
Ancillary Data in SMPTE-2038 PID ...................................................................................................... 4
VSF TR-01 in a TS .................................................................................................................................. 6
Supported Sources .............................................................................................................................. 6
Setting Up a New Event ...................................................................................................................... 6
Migrating from SDI Inputs to TR-01 Inputs .......................................................................................... 7

Outputs ............................................................................................................................................ 8
Reliable TS Output Group .................................................................................................................. 8
Output to MediaConnect .................................................................................................................... 8
Assumptions ......................................................................................................................................... 9
Setup Procedure .................................................................................................................................. 9
How It Works at Runtime .................................................................................................................... 12

Output to MediaStore ........................................................................................................................ 13
Step A: Set up AWS Elemental Live in AWS Identity and Access Management ......................... 13
Step B: Create the AWS Elemental Live Output Group ................................................................. 15

Video .................................................................................................................................................. 16
Working with HDR Color Space ........................................................................................................ 16
Working with the Input ......................................................................................................................... 17
Working with the Output ....................................................................................................................... 19
Location of HDR Fields on the Web Interface .................................................................................... 21
Location of HDR Fields in the XML .................................................................................................... 21
Dolby Vision HDR10 ............................................................................................................................ 22
QVBR and Rate Control Mode ............................................................................................................ 23
Handling Video Quality ....................................................................................................................... 25
Image Processing Controls ................................................................................................................ 25
Encoding Controls .............................................................................................................................. 33

Audio ................................................................................................................................................ 46
Dolby Metadata .................................................................................................................................. 46
Categories of Metadata ....................................................................................................................... 46
Source of Metadata ............................................................................................................................. 47
Impact on Output Audio ....................................................................................................................... 47
Codec Combinations ........................................................................................................................... 47
Setting up ........................................................................................................................................... 48
Output with AC3 .................................................................................................................................. 51
Output with EAC3 ............................................................................................................................... 54

HLS Rendition Groups ....................................................................................................................... 57
How Video Is Associated with Audio Rendition Groups ................................................................. 58
Rules for Rendition Groups ................................................................................................................. 58
Examples ............................................................................................................................................ 59
Creating HLS Rendition Groups ......................................................................................................... 61
Sample HLS Output Group ................................................................................................................ 73

AWS Elemental Live Features .......................................................................................................... 75
Working with Captions ....................................................................................................................... 75
Supported Features ............................................................................................................................ 75
Typical Scenarios ............................................................................................................................... 79
Setting Up for Captions ...................................................................................................................... 81
Examples of Implementing Use Cases .............................................................................................. 102
What Is AWS Elemental Live?

AWS Elemental Live is a real-time video service that lets you create live outputs for broadcast and streaming delivery.

You use AWS Elemental Live to transform live video content from one format and package into other formats and packages. You typically need to transform the content in order to provide a format and package that a playback device can handle. Playback devices include smartphones and set-top boxes attached to televisions.

Topics
• Finding Information About AWS Elemental Live (p. 1)
• Interfaces for AWS Elemental Live (p. 1)
• How AWS Elemental Live Works (p. 2)
• AWS Elemental Live Terminology (p. 2)

Finding Information About AWS Elemental Live

Information about AWS Elemental Live is available in the following guides and resources.

This Guide
This guide offers conceptual and procedural information for specific features of AWS Elemental Live. For a list of the features covered in this guide, see the topic list previous to this section.

AWS Elemental Live API Guide
This guide is intended for system integrators and AWS Elemental Live operators. It contains the following information:

• An outline of the interfaces for machine and human control, configuration, and monitoring. For a summary of the interfaces covered in the guide, see the next section.
• An overview of how to work with transcoding events, event profiles, and presets.
• A list and explanation of event and system parameters.

This guide is available on the Support tab of the web interface of your AWS Elemental Live appliance.

AWS Elemental Live Installation Guide
AWS Elemental Live Upgrade Guide
AWS Elemental Live Configuration Guide

Interfaces for AWS Elemental Live

AWS Elemental Live can be controlled, configured, and monitored through the following interfaces.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web browser via HTML</td>
<td>Using a web browser is the easiest way to control, configure, and monitor AWS Elemental Live. This interface is used when a human is interacting with</td>
</tr>
</tbody>
</table>

Version 2.17
1
How AWS Elemental Live Works

From the point of view of AWS Elemental Live, a live streaming workflow that includes AWS Elemental Live involves three systems:

- An AWS Elemental Live event, which ingests and transcodes source content.
- One or more upstream systems that provide the source content (the video) to AWS Elemental Live.

Examples of an upstream system are a streaming camera or appliance that is directly connected to the internet, or a contribution encoder that is located in a sports stadium where a sports event is being held.

The source content is in a specific package format and protocol. For example, the source content might be available as streaming HLS or streaming TS (transport stream). The source content contains video, audio, and optional captions streams that are in specific codecs or formats.

- One or more downstream systems that are the destinations for the output that AWS Elemental Live produces.

A typical downstream system consists of an origin service or a packager that is connected to AWS Elemental Live, a content distribution network (CDN) that is downstream of the origin service or the packager, and a playback device or website where the users view the content.

To create an AWS Elemental Live workflow, you create an event. Broadly speaking this event contains two sets of configuration information:

- A list of inputs (sources) and information about how to ingest those sources.
- A list of output groups that specifying packaging and encoding information.

To start processing the content, you start the event. When the event is running, it ingests the source content from the upstream system that is identified by the input. The event then transcodes that video (and the related audio, captions, and metadata) and creates outputs. AWS Elemental Live sends the outputs to the specified downstream systems.

AWS Elemental Live Terminology

CDN

A content distribution network (CDN) is a network of servers that is downstream of the origin server or packager. The CDN distributes the content from the origin server to dozens or hundreds
of networked servers that serve the content to your viewing users. This distributed network ensures that content can be delivered to thousands or millions of viewing users simultaneously.

**Downstream system**

The *downstream system* is a set of one or more servers that is positioned after AWS Elemental Live in the workflow. The downstream system handles the content that is output from AWS Elemental Live.

**Encode**

An encode exists within an output. There are three types of encodes: video, audio, and captions. Each encode contains the instructions for one video stream, one audio stream, or one captions track that the transcoding process will create. Different encodes have different characteristics. For example, one video encode produced from the input might be high resolution while another is low resolution.

**Event**

An AWS Elemental Live event ingests and transcodes (decodes and encodes) source content from the inputs that are attached to that event, and packages the new content into outputs.

**Event configuration**

An AWS Elemental Live event configuration contains information about how the event ingests, transcodes, and packages content into output.

**Origin service**

An origin service might be part of the downstream system that is positioned after AWS Elemental Live in the workflow. It accepts the video output from AWS Elemental Live.

**Output**

An output exists within an output group. It is a collection of encodes that you want to handle as one set.

**Output Group**

An output group is a collection of outputs within the AWS Elemental Live event.

**Packager**

A packager might be part of the downstream system. It accepts the video output from AWS Elemental Live and repackages it. AWS Elemental MediaPackage is a packager.

**Playback device**

A playback device is the final component of the downstream system. It is the device that the people who are your audience use to view the video.

**Source content**

The video content that AWS Elemental Live transcodes. The content typically consists of video, audio, captions, and metadata.

**Upstream system**

The system that is in front of AWS Elemental Live in the workflow and that holds the source content. Examples of an upstream system are a streaming camera or appliance that is directly connected to the internet, or a contribution encoder that is located in a stadium at a sports event.
Working with Inputs

This chapter describes how to set up the different types of inputs that AWS Elemental Live supports.

Topics
- Interleave 2SI Inputs (p. 4)
- Ingesting Ancillary Data from SMPTE-2038 in an MPEG-2 TS (p. 4)
- Ingesting from VSF TR-01 in a TS (p. 6)

Interleave 2SI Inputs

AWS Elemental Live supports Interleave 2SI inputs. This feature requires a specific model of appliance set up with an 8-port SDI card. To support 4Kp50/60 encoding workflow, the appliance that will ingest the Interleave 2SI input must be a L68x series or L700 series. To support a workflow to encode 4Kp50/60 input to SD or HD output, the appliance that will ingest the Interleave 2SI must be any of the following: the L28x, L300 series, L48x, and L500 series.

To use this feature, do the following:

- Make sure that the 8-port SDI card or cards are installed in the appliance.

To configure an event for this input, set the Select Type field to Interleave 4K (HD-2SI). Make sure you do not set the field to Interleave 4K when the input is Quadrant 4K, or vice versa. The appliance will detect the input but this mismatched configuration will result in readily observable video issues.

The Interleave 4K (HD-2SI) setting is applied to ports 1-4, 5-8, 9-12, or 13-16.

Media Info will report the input as either 2SI or SDI.

Ingesting Ancillary Data from SMPTE-2038 in an MPEG-2 TS

AWS Elemental Live supports extraction of select SDI ancillary data from SMPTE-2038 contained in MPEG-2 transport stream (TS) sources.

This procedure describes how to set up an event where the input is a TS that includes a SMPTE-2038 PID. This procedure applies for both TS sources that contain TR-01-compliant video and audio, and TS sources that contain video and audio that is not TR-01 compliant.

The ancillary data wrapped in a SMPTE-2038 can include any SDI ancillary data. AWS Elemental Live supports the ability to extract and process captions, timecode, AFD, and SCTE-104 messages.

Well-formed SMPTE-2038 Source

For AWS Elemental Live to extract and process the ancillary data appropriately, the SMPTE-2038 must meet certain criteria:
• The SMPTE-2038 must be present in every PMT.
• The PID in which the SMPTE-2038 is located must not change in the stream. There is no support for changing the PID and sending a new PMT identifying that PID.
• The stream should contain the SMPTE-2038 in only one PID. If it is present in more than one PID, there is no guarantee that AWS Elemental Live will identify the PID that appears first. It could choose another PID, with results you do not intend.

Set up a New Event for SMPTE-2038

Do the following to set up a new event for SMPTE-2038:

1. In Input, make sure you have set the Input type field to one of: Network Input, File Input, HLS File Input, or HLS Network Input (all these values are valid for MPEG-2 TS input). With any of these options, the Prefer SMPTE-2038 field appears in the Advanced section.
2. In Input > Advanced, set Prefer SMPTE-2038 to control how AWS Elemental Live ingests the SMPTE-2038 ancillary data.
   • If you do not want AWS Elemental Live to look at the SMPTE-2038, uncheck this field. When the Prefer SMPTE-2038 field is unchecked (default), then AWS Elemental Live follows the legacy behavior. It looks for ancillary data in the native TS. Even if a SMPTE-2038 PID is present, AWS Elemental Live ignores that PID.
   • If you want to use the SMPTE-2038 ancillary data, and assuming the stream contains well-formed SMPTE-2038 (above), then check this field.
3. Set the Timecode Source (in Video Selector) in the usual way:
   • If you checked Prefer SMPTE-2038, then set the source to Embedded in order to use the timecode in the SMPTE-2038. Or choose another value in order to use a different timecode source.
   • If you did not check Prefer SMPTE-2038, then set the source in the usual way. If you choose Embedded, the timecode will be extracted from the native TS (never from the SMPTE-2038).
4. In Input > Caption Selector, set up the captions in the usual way. For more information on working with captions, see Working with Captions (p. 75).

   The captions will be extracted as specified in the Prefer field, above.
5. If you need support for extracting OP-47 metadata (rather than captions) carried in SMPTE-2031, contact AWS Elemental Support through your company’s Private Space in AWS Elemental User Community.

   In Input > Timecode Configuration, set up the timecode for all outputs in the event in the usual way. This field is not affected by the presence or absence of SMPTE-2038.
6. In Output > Video Stream > Advanced set the Respond to AFD field and the Insert AFD signaling field in the usual way. These fields are not affected by the presence or absence of SMPTE-2038.
7. Set up the remainder of the event in the usual way.

How Assets and Ancillary Data Are Processed

If ancillary data is extracted from the SMPTE-2038 source, it is associated with the closest video frame. Regardless of where ancillary data is extracted from (a SMPTE-2038 PID or the native TS), once it is extracted, it is processed as follows:

• Captions are processed in the usual way, as specified in the output sections of the event.
• AFD (active format descriptors) is processed as specified by two fields in the Output > Video Stream > Advanced area: The Respond to AFD field and the Insert AFD signaling field.
• Time code is inserted as specified in the Timecode Configuration field.
• SCTE 104 markers are processed in the usual way: they are converted to SCTE 35 markers.
Ingesting from VSF TR-01 in a TS

AWS Elemental Live supports VSF TR-01 ingest over a transport stream (TS). Support for TR-01 provides a solution to migrating from SDI infrastructure to an all-IP infrastructure. Handling of IP inputs using TR-01 is almost feature-equivalent to the handling of SDI inputs.

The presence of TR-01 means that the video and audio are TR-01-compliant, as described below. In addition, SMPTE-2038 is required in a TR-01-compliant TS.

Supported Sources

Video in TR-01

The video source must be encoded with JPEG 2000 and can be one of the following:

<table>
<thead>
<tr>
<th>Definition</th>
<th>Resolutions</th>
<th>Maximum Bitrate</th>
<th>Color Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD</td>
<td>576i/25 and 480i/29.97</td>
<td>200 MBps</td>
<td>ITU-R BT 601-6</td>
</tr>
<tr>
<td>HD-SDI</td>
<td>Up to 1080i/25/29.97</td>
<td>200 MBps</td>
<td>ITU-R BT 709-5</td>
</tr>
<tr>
<td></td>
<td>and 720p/50/59.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3G-SDI</td>
<td>1080p/50 and 1080p/59.94</td>
<td>200 MBps</td>
<td>ITU-R BT 709-5</td>
</tr>
</tbody>
</table>

Audio Codecs in TR-01

The audio source can contain up to eight audio signals. Each audio signal is wrapped in SMPTE-302. At any given point in the source, the SMPTE-302 can contain:

- One AES3 signal pair of Dolby AC3. The AES3 pair is wrapped in SMPTE-337.
- One AES3 signal pair of Dolby E-AC3. The AES3 pair is wrapped in SMPTE-337.
- One AES3 signal pair of Dolby E. The AES3 pair is wrapped in SMPTE-337. There is support for multi-program Dolby E streams, in which case individual programs can be selected
- Or one uncompressed PCM audio stream.

There is support for source monitoring of audio codecs carried in SMPTE-302. If the audio codec (or type) changes, AWS Elemental Live detects the change and processes the new codec appropriately.

AWS Elemental Live enforces the rule about the contents of the SMPTE-302 as follows:

- For the audio codecs other than PCM, if the SMPTE-302 contains invalid content (there is more than one signal pair in the SMPTE-302), then only the first signal pair will be processed and the remaining signal pairs will be discarded.
- For PCM, if the SMPTE-302 contains invalid content (there is more than one signal pair in the SMPTE-302), then AWS Elemental Live will process both. The probable result will be garbled audio.
- If the audio source contains an unsupported codec or format, AWS Elemental Live logs a message, displays an alert on the web interface, and inserts silence. AWS Elemental Live continues monitoring the source and detecting the format.

Setting Up a New Event

Perform the following steps to set up a new event:
1. In Input, set the Input type to Network Input. In Network Location, specify an IP address.

2. In Input > Advanced, set the Prefer SMPTE-2038 field as appropriate:
   - If you do not want AWS Elemental Live to look at the SMPTE-2038, uncheck this field. When the Prefer SMPTE-2038 field is unchecked (default), then AWS Elemental Live follows the legacy behavior. It looks for ancillary data in the native TS. Even if a SMPTE-2038 PID is present, AWS Elemental Live ignores that PID.
   - If you want to use the SMPTE-2038 ancillary data, and assuming the stream contains well-formed SMPTE-2038 (above), then check this field.

   For information about how SMPTE-2038 ancillary data is extracted and processed, see Ingesting Ancillary Data from SMPTE-2038 in an MPEG-2 TS (p. 4).

3. In Input > Video Selector and Input > Audio Selector, set up the video and audio in the usual way. There are no special steps for setting up TR-01 video or audio inputs.

4. Set up the remainder of the event in the usual way.

The video identified in the video selector and the audio identified in the audio selector are extracted from the TR-01 in the TS. The video and audio are processed in the usual way, as specified in the output sections of the event. The ancillary data (if used) is extracted from the SMPTE-2038 and processed as described in Ingesting Ancillary Data from SMPTE-2038 in an MPEG-2 TS (p. 4).

**Migrating from SDI Inputs to TR-01 Inputs**

If you have an event that is configured to use an SDI input, you can easily adapt it to use an IP network feed that contains TR-01-compliant video and audio. Change the existing event as follows:

- Change the **Input Type** to **Network Input**. In **Network Location**, specify an IP address.
- Set the **Prefer SMPTE-2038** box as appropriate:
  - If you do not want AWS Elemental Live to look at the SMPTE-2038, uncheck this field. When the **Prefer SMPTE-2038** field is unchecked (default), then AWS Elemental Live follows the legacy behavior. It looks for ancillary data in the native TS. Even if a SMPTE-2038 PID is present, AWS Elemental Live ignores that PID.
  - If you want to use the SMPTE-2038 ancillary data, and assuming the stream contains well-formed SMPTE-2038 (above), then check this field.

   For information about how SMPTE-2038 ancillary data is extracted and processed, see Ingesting Ancillary Data from SMPTE-2038 in an MPEG-2 TS (p. 4).

There is no requirement to change any other fields: the event should now perform the same processing as previously.

The video identified in the video selector and the audio identified in the audio selector are extracted from the TR-01 in the TS. The video and audio are processed in the usual way, as specified in the output sections of the event. The ancillary data (if used) is extracted from the SMPTE-2038 and processed as described in Ingesting Ancillary Data from SMPTE-2038 in an MPEG-2 TS (p. 4).
Working with Outputs

This chapter describes how to set up the different types of outputs that AWS Elemental Live supports.

**Topics**
- Delivering TS Output Using the Zixi Protocol (p. 8)
- Setting up AWS Elemental Live as a Contribution Encoder for AWS Elemental MediaConnect (p. 8)
- Sending AWS Elemental Live Output to AWS Elemental MediaStore (p. 13)

**Delivering TS Output Using the Zixi Protocol**

You can deliver TS output using the Zixi protocol. The destination is considered to be a "Zixi broadcaster".

1. In the AWS Elemental Live event, go to Output Groups > Reliable TS.
2. Choose Add Output to create an output in this Reliable TS output group.
3. Complete the fields in each output as follows:
   - **Delivery Protocol**: Zixi
   - **Destination/Amazon Resource Name**: Enter the IP address (that you obtained from the Zixi broadcaster), a colon, and the port (default is 2088). For example:
     
     ```
     zixi://198.51.100.0:2088
     ```
   - **Interface**: Optional. See the tooltip.
   - **If the Zixi broadcaster requires that you authenticate, obtain the username and password from them. Then choose the Lock icon and complete the fields that appear.**
     - **Username/Access Key ID**: The username.
     - **Password/Secret Access Key**: The password.
   - **Stream ID**: Obtain this value from the Zixi broadcaster. It is required.
   - **Latency**: See the tooltip.
   - **Encryption**: Choose None, or choose an algorithm. The downstream system must support the algorithm you choose.
   - **Key Value**: Enter a SHA-256 encryption key that you have obtained from the Zixi broadcaster

Repeat the preceding steps to create a second output in this output group, if applicable. Use the same user name and password. You can also use the same encryption key, if you are encrypting.

**Setting up AWS Elemental Live as a Contribution Encoder for AWS Elemental MediaConnect**

You can set up an AWS Elemental MediaConnect flow as the output from AWS Elemental Live. In this setup, AWS Elemental Live is the contribution encoder for a MediaConnect flow. You can choose to encrypt the output during delivery to MediaConnect.

**Topics**
- Assumptions (p. 9)
Assumptions


This article assumes that you have already set up permissions for MediaConnect. So you have created at least one AWS user and given permissions to those users so that they can use the features of MediaConnect. Specifically, for the purposes of this procedure, the user can create a MediaConnect flow. You have also set up MediaConnect as a trusted entity with Secrets Manager; see Step 3: Create an IAM Role with a Trusted Relationship in the AWS Elemental MediaConnect User Guide.

This article does not assume that you have set up AWS Elemental Live with permissions in AWS. Setting up those permissions is one of the steps in this article.

Setup Procedure

Step A: Create a Role in IAM and Attach Policies

You must use AWS Identity and Access Management (IAM) to set up AWS Elemental Live as an AWS user (the "AWS Elemental Live user") and give it permissions so that it can communicate with AWS Secrets Manager and AWS Elemental MediaConnect. You must:

• Create policies that contain specific permissions.
• Create the "AWS Elemental Live user" in your AWS account. The user must be in the same AWS account as the user who is operating MediaConnect.
• Associate the AWS Elemental Live user with those policies, which gives the user the permissions of those policies.

Create a Policy for AWS Elemental Live to Make Requests to MediaConnect

AWS Elemental Live must have permissions on MediaConnect. Follow this procedure to set up these permissions:

To create a policy for AWS Elemental Live to make requests to MediaConnect

1. Log into the AWS console and go to the IAM console.
2. On the left menu, choose Policies. Use the filters to determine if there is already a policy with a name similar to "ElementalAccessToMediaConnect".
3. If the policy does not exist, choose Create policy. Click the Visual editor tab and create the policy using the IAM policy generator. This generator lets you choose the service from a list and then choose operations from a list:
   • Service: MediaConnect.
   • Resources: If your organization does not have strict rules about accessing containers on MediaConnect, you can ignore this section; you will have access to all flows. Otherwise, follow your internal policies to identify specific flows.
   • Give the policy a name such as "ElementalAccessToMediaConnect"
For more information on how to create and manage IAM policies, see the IAM User Guide.

Create a Policy for AWS Elemental Live to Make Requests to Secrets Manager

If you plan to encrypt the output from AWS Elemental Live when you send it to MediaConnect, then AWS Elemental Live must have permissions on AWS Secrets Manager. Follow this procedure to set up these permissions:

**To create a policy for AWS Elemental Live to make requests to Secrets Manager**

1. Log into the AWS console and go to the IAM console. Choose Policies and look for a policy that gives MediaConnect the permissions for Secrets Manager. If you or someone else previously followed the procedure in Step 2: Create an IAM Policy to Allow AWS Elemental MediaConnect to Access Your Secret, then this policy will be called `SecretsManagerForMediaConnect`.
2. If this policy exists, make sure it contains the following actions:
   - DescribeSecret
   - GetResourcePolicy
   - GetSecretValue
   - ListSecretVersionIds
3. Also make sure that the resources section gives access to the ARN of the secret that you will use. Read the information in IAM Policy Examples for Secrets in AWS Secrets Manager. You may need to edit the policy to include the ARN for this secret in the resources section.
4. If the policy does not exist, follow the procedure in Step 2: Create an IAM Policy to Allow AWS Elemental MediaConnect to Access Your Secret to create the policy.

Create a User

**To create a user**

1. Log into the AWS console and go to the IAM console.
2. If the user does not exist or it does exist but you want to create separate users for each Elemental product, choose Add User. (Note that you may want separate users for separate products, but there is probably no need to create a separate user for each Elemental node.) Follow the prompts to add the user with this information:
   - Give the user a name such as `ElementalUser`.
   - For Access type, choose Programmatic access. Do not choose Console access.
   - In permissions, choose Attach existing policies directly. Attach the policies you created above. For example, `ElementalAccessToMediaConnect` and `SecretsManagerReadSecrets`.
   - Ignore tags.
3. Create the user and choose Close.
4. Choose the user by clicking the name, for example, click ElementalUser.
5. Choose the Security tab.
6. Click **Create Access Key**.
7. On the Create access key dialog, choose to download the .csv file. Save the file in a safe place, so that you have a permanent record of the access key ID and the secret access key.
   - The Access key ID looks like this: AKIAIOSFODNN7EXAMPLE
   - The Secret access key looks like this: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
8. Give the Access key ID and the Secret access key to the AWS Elemental Live operator. Do not give the username and password to the operator.
9. How it works: You have created an AWS user with the permissions required to make requests to MediaConnect and optionally to Secrets Manager. When the AWS Elemental Live user sets up the output with MediaConnect as the destination, they will enter the Access key ID and Secret access key. When the AWS Elemental Live event is running, AWS Elemental Live sends these two IDs to the AWS services, instead of sending the user name and password. These IDs provide authorization for the AWS Elemental Live node to make requests to the AWS services.

Step B: Set up for Encryption (Optional)

If you are encrypting the AWS Elemental Live output, you must generate an encryption key and set it up in Secrets Manager. In this scenario, Secrets Manager is effectively acting as the key server for the encryption key. Secrets Manager serves the key to AWS Elemental Live so it can encrypt and to MediaConnect to it can decrypt. In this scenario, encryption/decryption is supported with a static SHA-256 encryption key and using the AES-256, AES-192, or AES-128 algorithm.

Perform the following steps according to the security policies and procedures for your organization.

To set up for encryption

1. Use a suitable tool for generating a SHA-256 encryption key from a seed that you specify. AWS does not provide a generation tool. Note that you need only one key, even if you are creating two flows.
2. Save the key to the secret, as described in Setting Up Static Key Encryption Using AWS Elemental MediaConnect. You must assign a name to the secret, for example, "key_sports". Save the key in the same AWS Region as the flow you plan to create.
3. Make a note of the ARN for this secret. You need this ARN when you create the MediaConnect flow. It looks like the following example, where "key_sports" is the name you assigned to the secret.

   arn:aws:secretsmanager:us-west-2:111122223333:secret:key_sports-7g8H9i

Step C: Create the AWS Elemental MediaConnect Flows

You must follow this procedure before you create the AWS Elemental Live outputs because AWS Elemental Live needs data that is generated by this procedure.

To create the MediaConnect flows

1. Create one or two MediaConnect flows. (Create two flows if your have set up AWS Elemental Live for output redundancy using output locking. If you have not set up redundant outputs, create one flow.)
2. Follow the procedure in Creating a Flow in the AWS Elemental MediaConnect User Guide.
3. Complete Availability Zone and Name as appropriate. These fields do not relate to using AWS Elemental Live as the source.
4. In the Source section, follow the steps for setting up a standard source. Specifically:
   - Protocol: Zixi push.
   - Whitelist CIDR block: This is the IP address (in CIDR format) of the Elemental node that will be delivering to this flow. It must be a public facing IP address. Speak to your organization's administrator for a value to enter here.
   - Stream ID: You must enter a value when AWS Elemental Live is the source.
5. If you are encrypting the video, check Enable in the Decryption section and complete the fields as described in the MediaConnect documentation. Specifically:
   - Decryption type: Always Static key.
   - Role ARN: The role that has been set up for MediaConnect to be a trusted entity with Secrets Manager. See Step 3: Create an IAM Role with a Trusted Relationship in the AWS Elemental
MediaConnect User Guide. You must specify this role ARN here so that MediaConnect can obtain the encryption key.

- To find the ARN for the role, go to the IAM console, choose Roles, click the name of the role, and look at the Role ARN field in the Summary. The role ARN looks like this:

  arn:aws:iam::111122223333:role/MediaConnectASM

- Secret ARN: The ARN you obtained in step A, for example:

  arn:aws:secretsmanager:us-west-2:111122223333:secret:key_sports-7g8H9i

- Decryption algorithm: Specify the algorithm that you want to use. AWS Elemental Live will be instructed to use this algorithm to encrypt. MediaConnect will read this information and use this algorithm to decrypt.

6. When you create each flow, MediaConnect creates an ARN for that flow. The ARNs look like the following, where "curling_finals_A" and "curling_finals_B" are the flow names you specified in each flow:

   arn:aws:mediaconnect:us-west-1:111122223333:flow:1bgf67:curling_finals_A

7. Make a note of these ARNs. You need them to set up the AWS Elemental Live output(s).

Step D: Create the AWS Elemental Live Output Group

You must create one output group of type "Reliable TS". Inside that group, you must create one or two outputs: create two outputs if you created two MediaConnect flows, create one output if you created only one flow.

To create the AWS Elemental Live output group

1. In the AWS Elemental Live event, go to Output Groups > Reliable TS.
2. Click Add Output to create an output in this Reliable TS output group.
3. Complete the fields in each output as follows:

   - Delivery Protocol: Choose AWS Elemental MediaConnect.
   - Destination/Amazon Resource Name: Enter the ARN for the flow. Following from the example above, enter the following in the first output:

     arn:aws:mediaconnect:us-west-1:111122223333:flow:1bgf67:curling_finals_A

   - Interface: Optional; see the tooltip.
   - Lock icon: Click this icon. Two more fields appear:
     - Username/Access Key ID: The Access key ID for the user you created in AWS IAM. For example, AKIAIOSFODNN7EXAMPLE
     - Password/Secret Access Key: The Secret access key for this user. For example, wJalrXUtFEMIGK7MDENG/bPxRfyQEXAMPLEKEY

4. Note that there is no encryption field. See "How It Works at Runtime", below, to understand how encryption is handled.
5. Repeat these steps to create a second output in this output group, if applicable. Use the same Access key ID and Secret access key.

How It Works at Runtime

Here is the data that AWS Elemental Live has: The flow ARN. The Access key ID and Secret access key.

Here is the data that MediaConnect has: The flow ARN. The destination IPs and protocol details. The
encryption type and algorithm. The role ARN (for obtaining the secret - the encryption key). The secret ARN. When the event starts, AWS Elemental Live authenticates with AWS using the AWS access key ID and AWS secret access key. It then sends the flow ARN to MediaConnect. MediaConnect accepts the request because AWS Elemental Live has permission to make requests to MediaConnect. MediaConnect looks up the flow and determines if the flow is set up for encryption.

- If the flow is set up for encryption, MediaConnect sends the encryption type and algorithm information, and the secret ARN to AWS Elemental Live. AWS Elemental Live uses the secret ARN to get the secret (the encryption key) from Secrets Manager. Secrets Manager accepts the request from AWS Elemental Live because AWS Elemental Live has permission to get this secret.

AWS Elemental Live uses the encryption key to encrypt the video and sends the encrypted video to MediaConnect.

MediaConnect in its turn uses the secret ARN to get the secret (the encryption key) from the Secrets Manager. Secrets Manager accepts the request from MediaConnect because MediaConnect has permission to get this secret; it has permission because it has been set up as a trusted entity with Secrets Manager. MediaConnect uses the encryption key to decrypt the video.

- If the flow is not set up for encryption, MediaConnect instructs AWS Elemental Live to deliver the video unencrypted. Secrets Manager is not involved.

### Sending AWS Elemental Live Output to AWS Elemental MediaStore

You can set up a container on AWS Elemental MediaStore as the destination for Apple HLS and DASH outputs.

This article assumes you know how to use the AWS Management Console and AWS Identity and Access Management, and that you have access to IAM User Guide.

This article assumes that you or someone in your organization has created the MediaStore container where AWS Elemental Live will deliver the output. Make sure you have the path to this container or containers.

**Step A: Set up AWS Elemental Live in AWS Identity and Access Management**

You must use the IAM (AWS Identity and Access Management) service to set up AWS Elemental Live as an AWS user (the "Elemental user") and give it permissions so that it can communicate with MediaStore. You must do the following:

- Create a policy that contains specific permissions.
- Create the AWS Elemental Live user in your AWS account. The user must be in the same AWS account as the user who is operating AWS Elemental MediaStore.
- Associate the AWS Elemental Live user with the policy, which gives the user the permissions of that policy.

You perform this setup only once. You can use the same "Elemental user" every time you want to send output to MediaStore.

**Create a policy for AWS Elemental Live to Make Requests to MediaStore**
AWS Elemental Live User Guide

Step A: Set up AWS Elemental Live in
AWS Identity and Access Management

AWS Elemental Live must have permissions on MediaStore. Follow this procedure to set up these permissions:

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. On the left menu, choose Policies. Use the filters to determine if there is already a policy with a name similar to ElementalAccessToMediaStore.
3. If the policy does not exist, choose Create policy. Choose the Visual editor tab and create the policy using the IAM policy generator. This generator lets you choose the service from a list and then choose operations from a list:
   - Service: MediaStore
   - Actions: Under List, choose DescribeContainer
   - Actions: Under Read, choose GetObject, DescribeObject, GetContainerPolicy
   - Actions: Under Write, choose PutObject.
   - Resources: If your organization does not have strict rules about accessing containers on MediaStore, you can ignore this section; you will have access to all containers. Otherwise, follow your internal policies to identify specific containers.
   - Give the group a name such as ElementalAccessToMediaStore.

For detailed instructions on creating a policy, see IAM User Guide Creating IAM Policies.

Create a User

Follow this procedure to create a user:

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. On the left menu, choose User. Use the filters to determine if there is already a user for Elemental products. The user might be called ElementalUser.
3. If the user does not exist or it does exist but you want to create separate users for each Elemental product, choose Add User. (Note that you may want separate users for separate products, but there is probably no need to create a separate user for each Elemental node.) Follow the prompts to add the user with this information:
   - Give the user a name such as ElementalUser.
   - For Access type, choose Programmatic access. Do not choose Console access.
   - In permissions, choose Attach existing policies directly. Attach the policy you created above. For example, ElementalAccessToMediaStore.
   - Ignore tags.
4. Create the user and choose Close.
5. On the left menu, choose Users again:
   - Choose the user name, for example, ElementalUser.
   - Choose the Security tab.
   - Choose Create Access Key.
   - On the Create access key dialog, choose to download the .csv file. Save the file in a safe place, so that you have a permanent record of the Access key ID and the Secret access key.

The Access key ID looks like this: KIAIOSFODNNYEXAMPLE

The Secret access key looks like this: 94afd1f2e64d908bc90dcbca0035a5b567EXAMPLE
6. Give the Access key ID and the Secret access key to the AWS Elemental Live operator. Do not give the username and password to the operator.

This creates an AWS user with the permissions required to let AWS Elemental Live make requests to MediaStore. When the AWS Elemental Live operator sets up the output with MediaStore as the
destination, they will enter the **Access key ID** and **Secret access key**. When the AWS Elemental Live event is running, AWS Elemental Live sends these two IDs to the AWS service, instead of sending the user name and password. These IDs provide authorization to AWS for the AWS Elemental Live node to make requests to MediaStore.

**Step B: Create the AWS Elemental Live Output Group**

To set up MediaStore as the destination in the HLS or DASH output group:

1. Obtain the endpoint for the MediaStore container where you want to send the output. For more information, see [AWS Elemental MediaStore User Guide Viewing the Details for a Container](https://aws.amazon.com/documentation/mediastore). The following is an example of an endpoint:

   ```
   https://w9710g.data.mediastore.us-west-2.amazonaws.com
   ```

2. In the AWS Elemental Live event, go to the HLS or DASH output group and in **Output > HTTP Push Dialect** field, choose **AWS Elemental MediaStore**.

   For details on these fields, see the field tooltips.

4. In the output group > **Output > Destination** field (above the HTTP Push Dialect field), enter the destination in the format `<endpoint>/path/file/`. For example:

   ```
   https://w9710g.data.mediastore.us-west-2.amazonaws.com/sports/curling/
   ```

   where `https://w9710g.data.mediastore.us-west-2.amazonaws.com/` is the endpoint for the MediaStore container and `/sports/curling/` is the name of the MediaStore object.

5. Choose the **Lock** icon. Two more fields appear.

   - **Username/Access Key ID**: The Access key ID you created in IAM. For example, `AKIAIOSFODNNYEXAMPL`
   - **Password/Secret Access Key**: The Secret access key you created in IAM. For example, `wJalrXUtntFEMI/K7MDENG/bPxrYtCYEXAMPLEKEY`

   Repeat the preceding steps to create a second output in this output group, if applicable.
Working with Video

This chapter describes how to set up the video features of AWS Elemental Live.

**Topics**
- Working with HDR Color Space (p. 16)
- Dolby Vision HDR10 (p. 22)
- Setting up QVBR and Rate Control Mode (p. 23)
- Handling Video Quality (p. 25)

## Working with HDR Color Space

The input video may include three sets of data:

- Color space used for the range of pixel colors supported.
- Brightness function used for the pixel. Also known as gamma tables, look-up tables (LUT), Electro-optical Transfer Function (EOTF), and transfer function.
- Display metadata.

In outputs that are encoded using the HEVC (H.265) codec, AWS Elemental Live can create content that adheres to the following standards.

<table>
<thead>
<tr>
<th></th>
<th>HDR10</th>
<th>SMPTE ST 2084 (PQ)</th>
<th>HLG</th>
<th>HDR10 rec. 2020</th>
<th>SDR BT. 709</th>
<th>SDR BT. 601</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color Space</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Not applicable</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Brightness</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Not applicable</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Display Metadata</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Not applicable</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

AWS Elemental Live can work with HDR10 and HLG 2020 data from specific inputs:

- A UDP network source.
- An SDI source.
• A ProRes 4444 XQ file source.

Topics
• Working with the Input (p. 17)
• Working with the Output (p. 19)
• Location of HDR Fields on the Web Interface (p. 21)
• Location of HDR Fields in the XML (p. 21)

Working with the Input

In each input, you can specify how you want AWS Elemental Live to work with the data in the input.

• Option 1. Follow the data in the input (p. 17)
• Option 2. Override the data throughout (p. 17)
• Option 3. Override the data where necessary (p. 18)

These options have an effect only in outputs where the video is being encoded with HEVC. With other outputs (those encoded using MPEG-2 or H.264 for example), this data is never included in the output.

Input Option 1. Follow the Data in the Input

By default, AWS Elemental Live follows the color space data that is in the input and retains it for inclusion in the output. In this case, if the color space changes during ingestion (for example, from SDR 601 to SDR 709), then AWS Elemental Live detects the change.

This handling is appropriate if the color space data is always present in the input and is always accurate. Historically, SDR 601 frequently was not indicated in the input: the input was color and there was no other choice. Even with SDR 709 input, the color space may not be explicitly indicated. However, with HDR content and with 4K content, the color space is nearly always specified in the content.

It is also appropriate if you know that the color space data is completely inaccurate and you simply want to strip it out of the output: you set to Follow when setting up the input, and then you will explicitly remove it when setting up the output.

In the input, set up the color space:

• Color space: Follow

Input Option 2. Override the Data Throughout

If the specified color space is HDR10, you can configure AWS Elemental Live to modify the color space data in the entire input to a color space you specify and to Mastering Display data you specify.

It is appropriate to override the data throughout if you know that the input uses the same color space throughout but that, in some portions of the video, the color space data is either missing or wrong. For example, the input is all SDR 601, but some portions are marked as SDR 709.

In the input, set up Override Throughout as follows:

• Color Space : A value other than Follow.
• Force Color : Checked.
• Mastering Display Information: These fields appear only if you have set the Color space to HDR10. For help, see the tooltips for each field on the AWS Elemental Live web interface.
If you do not plan to convert the color space in the output, then set the fields appropriately. Make sure to obtain values used in the color grading process for the input; you cannot use the defaults or null values and expect to obtain valid color results. These values are included in all the color space data in the input.

If you do plan to convert the color space, these values are irrelevant; leave the defaults.

**Input Option 3. Override the Data Where Necessary**

You can configure AWS Elemental Live to modify the color space data for portions of the input to retain existing color space data. But for portions that do not include color space data, the input is tagged for the color space you specify.

This handling is appropriate if you know that the input has some missing color space data, but where it does have color space data, that data is correct.

In the input, set up as follows:

- **Color Space**: A value other than "Follow."
- **Force Color**: Unchecked.
- **Mastering Display Information**: These fields appear only if **Color Space** is set to HDR10. For help, see the tooltips for each field on the AWS Elemental Live web interface.

If you do not plan to convert the color space in the output, then set the fields appropriately. Make sure to obtain values used in the color grading process for the input; you cannot use the defaults or null values and expect to obtain valid color results. These values are included whenever the color space data is being modified.

If you do plan to convert the color space, these values are irrelevant; leave the defaults.

**Tips for HDR Master Display Information**

**Red, Green, Blue, White Point X and Y**

Your color grader may provide numbers like this for X and Y points:

- G (x=0.265, y=0.690)
- B (x=0.150, y=0.060)
- R (x=0.680, y=0.320)

You must convert these numbers to numbers like this:

- G (13250, 34500)
- B (7500, 3000)
- R (34000, 16000)

To convert between the two formats, divide each number by 0.00002 as per the HEVC specification:

For example, 0.265 divided by 0.00002 is 13250.

**Max/Min Luminescence**

The maximum and minimum luminance are given in units of **0.0001 candelas per square meter**. Your color grader may provide this value in candelas per square meter instead. If so, then convert these numbers by multiplying by 10,000, then entering the result in the UI.
Working with the Output

AWS Elemental Live supports passthrough, conversion, and removal of color space information from the video input to the output.

In each video encode in each output, you can specify how you want AWS Elemental Live to work with the existing color space data.

- Option 1. Pass through the color space specified in the input (p. 19)
- Option 2. Convert the color space (p. 19)
- Option 3. Remove the color space data from the output (p. 20)

Output Option 1. Pass through the Color Space

In the HEVC video encodes, set up as follows:

- **Insert Color Metadata**: Checked, to include metadata.
- **Color Space Conversion**: None.

Output Option 2. Convert the Color Space and Brightness Function

You can convert the color space and brightness function used so that the effect in the new standard is the same as the effect in the old standard.

**Supported Conversions**

Color space and brightness data from the input can be converted in the output as follows:

<table>
<thead>
<tr>
<th>Converting from</th>
<th>To SDR 601</th>
<th>To SDR 709</th>
<th>To HLG 2020</th>
<th>To HDR10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>To SDR 601</strong></td>
<td>Yes. See Case A below.</td>
<td>Yes. See Case B below.</td>
<td>Yes. See Case C below.</td>
<td></td>
</tr>
<tr>
<td><strong>To SDR 709</strong></td>
<td>Not supported</td>
<td></td>
<td>Yes. See Case B below.</td>
<td>Yes. See Case C below.</td>
</tr>
<tr>
<td><strong>To HLG 2020</strong></td>
<td></td>
<td>Yes. See Case B below.</td>
<td></td>
<td>Yes. See Case C below.</td>
</tr>
<tr>
<td><strong>To HDR10</strong></td>
<td></td>
<td>Not supported</td>
<td></td>
<td>Yes. See Case D below.</td>
</tr>
</tbody>
</table>

Conversion is performed as follows:

**Case A**

- Handling of color space: The video is tagged for the new color space, but there is no change to the pixel values.
- Handling of brightness: No change (the source and target use the same brightness function).
- Handling of metadata: Not applicable.
Case B

- Handling of color space: The video is tagged for the new color space, but there is no change to the pixel values.
- Handling of brightness: The position on the source function is mapped to the corresponding position on the target function; there is no scaling of values.
- Handling of metadata: Not applicable.

Case C

- Handling of color space: The video is tagged for the new color space, but there is no change to the pixel values (given that the source and target both use the same color space).
- Handling of brightness: The position on the source function is mapped to the corresponding position on the target function; there is no scaling of values. In this way, the artistic intent is retained.
- Handling of metadata: Metadata is added with default values.

Case D

- Handling of color space: The video is tagged for the new color space, but there is no change to the pixel values (given that the source and target both use the same color space).
- Handling of brightness: The position on the source function is mapped to the same position on the target function; there is no scaling of values. In this way, the artistic intent is retained.
- Handling of metadata: Not applicable.

Setup

In applicable HEVC video encodes, set up as follows:

- **Codec**: HEVC
- **Profile**: If you are passing through HDR10 or HLG2020 or converting to HDR10 or HLG 2020, choose one of the profiles with "Main10" in the name. If you are passing through or converting SDR, choose any profile that is appropriate.
- **Insert Color Metadata**: Checked.
- **Preprocessors > Color Corrector > Color Space Conversion**: One of the color space choices. If you choose HDR10, then HDR Mastering Display Information fields appear.
- **Preprocessors > Color Corrector > HDR Mastering Display Information**: This appears only if you choose to convert to HDR10. Leave the default values. Or, if you have previously converted similar content to HDR10 and you have been given better metadata to use with that content, then enter it here.

Note that the other fields in the Color Corrector preprocessor are not related specifically to HDR10.

Output Option 3. Remove the Color Space Data from the Output

Generally, choose this option only if you know the pixel data and color space data in the input is incorrect.

To remove color space data from the output of applicable HEVC video encodes, set up as follows:

- **Insert Color Metadata**: Unchecked.
### Location of HDR Fields on the Web Interface

- Input > Video Selector > Color Space
- Input > Video Selector > Force Color
- Input > Video Selector > HDR Master Display Information group of fields
- Stream > Video > Video Codec
- Stream > Video > Insert Color Metadata (near the top of the tab)
- Stream > Video > Profile (near the bottom of the tab)
- Stream > Video > Preprocessors > Color Corrector > Color Space Conversion

### Location of HDR Fields in the XML

<table>
<thead>
<tr>
<th>Input &gt; Video Selector &gt; Color Space</th>
<th>input/video_selector/color_space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input &gt; Video Selector &gt; Force Color</td>
<td>input/video_selector/force_color</td>
</tr>
</tbody>
</table>
| Input > Video Selector > HDR Master Display Information group of fields | input/video_selector/hdr10_metadata/
| | blue_primary_x |
| | input/video_selector/hdr10_metadata/
| | blue_primary_y |
| | input/video_selector/hdr10_metadata/
| | green_primary_x |
| | input/video_selector/hdr10_metadata/
| | green_primary_y |
| | input/video_selector/hdr10_metadata/
| | max_content_light |
| | input/video_selector/hdr10_metadata/
| | max_luminescence |
| | input/video_selector/hdr10_metadata/
| | max_picture_avg_light |
| | input/video_selector/hdr10_metadata/
| | min_luminescence |
| | input/video_selector/hdr10_metadata/
| | red_primary_x |
| | input/video_selector/hdr10_metadata/
| | red_primary_y |
| | input/video_selector/hdr10_metadata/
| | white_point_x |
| | input/video_selector/hdr10_metadata/
| | white_point_y |
| Stream > Video > Video Codec | stream_assembly/video_description/codec |
| Stream > Video > Insert Color Metadata | stream_assembly/video_description/
<p>| | insert_color_metadata |</p>
<table>
<thead>
<tr>
<th>Stream &gt; Video &gt; Profile</th>
<th>stream_assembly/video_description/profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream &gt; Video &gt; Preprocessors &gt; Color Corrector &gt; Color Space Correction &gt; Color Space Conversion</td>
<td>stream_assembly/video_description/color_corrector/color_space_conversion</td>
</tr>
<tr>
<td>Stream &gt; Video &gt; Preprocessors &gt; Color Corrector &gt; Color Space Correction &gt; HDR Master Display Information group of fields</td>
<td>stream_assembly/video_description/color_corrector/hdr10_metadata/blue_primary_x</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/video_preprocessors/color_corrector/hdr10_metadata/blue_primary_y</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/video_preprocessors/color_corrector/hdr10_metadata/green_primary_x</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/video_preprocessors/color_corrector/hdr10_metadata/green_primary_y</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/video_preprocessors/color_corrector/hdr10_metadata/max_content_light</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/video_preprocessors/color_corrector/hdr10_metadata/max_luminescence</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/video_preprocessors/color_corrector/hdr10_metadata/max_picture_avg_light</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/video_preprocessors/color_corrector/hdr10_metadata/min_luminescence</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/video_preprocessors/color_corrector/hdr10_metadata/red_primary_x</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/video_preprocessors/color_corrector/hdr10_metadata/red_primary_y</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/video_preprocessors/color_corrector/hdr10_metadata/white_point_x</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/video_preprocessors/color_corrector/hdr10_metadata/white_point_y</td>
</tr>
</tbody>
</table>

**Dolby Vision HDR10**

AWS Elemental Live supports ingest and output of Dolby Vision HDR10 content. The appliance must support SDI or Quad-SDI input. AWS Elemental Live continuously monitors the active source for the
presence of Dolby Vision RPU; whenever it detects them, it gracefully transitions to outputting an HEVC HDR10 signal with the proper Dolby Vision conditioning.

This feature requires a Dolby Vision player (TV or Set Top Box or other software player) to display the Dolby Vision content correctly. This feature was tested using AWS Elemental Live by generating Archive outputs of the video, stored on a USB, and played on an LG C7 TV. The output also plays on the TV as an HLS stream.

### Input and Output Setup

The source format may be up to 4K (UHD). An external Dolby HPU preprocessor must condition the HDR10 source content to include the RPU metadata, resulting in a Dolby Vision that can be input to AWS Elemental Live. For Quadrant 4K sources, the RPU metadata is carried in the first quadrant signal.

Set up the AWS Elemental Live event as follows:

- Set the **Input type** to **HD-SDI** or **Quadrant 4K**.
- In **Input > Advanced**, set the **Color Space** to **Follow**.
- In the **Output > Stream > Video** section, the output must be set up for the HEVC (H.265) codec and the **Profile** must be **Main10/Main** or **Main10/High**.

### How AWS Elemental Live Processes the Video

The event can be started at any time even if the Dolby conditioning preprocessor is not active. The presence or absence of Dolby RPU conditioning is constantly monitored at the input. If the Dolby preprocessor stops conditioning the active source, AWS Elemental Live detects the loss of RPU at the source and gracefully transitions to a non-Dolby output as described below. If the preprocessor resumes condition, AWS Elemental Live gracefully transitions to a Dolby output.

Content is processed as follows:

- Sources with the Dolby RPU are encode in the output as an HEVC HDR10 signal with the proper Dolby Vision conditioning (with the dynamic metadata and appropriate PMT conditioning). The stream can be up to 4K.
- Sources without Dolby conditioning are by default encoded in the same HEVC form: SDR sources are output as SDR, HDR10 as HDR10, and HLG as HLG.

### Logging

AWS Elemental Live provides the following related logs:

- When AWS Elemental Live transitions to an HEVC HDR10 signal with the proper Dolby Vision conditioning, the following message is written to the EME log:
  
  ```
  Muxer TS (job number) Dolby Vision RPU detected; conditioning initial PMT.
  ```

- When AWS Elemental Live transitions to a non-Dolby output, the following message is written to the EME log:
  
  ```
  Muxer TS (job number) Dolby Vision RPU no longer detected; removing PMT conditioning.
  ```

### Setting up QVBR and Rate Control Mode

The Rate Control Mode fields in the **Video** section of each output on the AWS Elemental Live web interface let you control the quality and bitrate of the video.
When encoding visually complex video (such as high-motion sports events with brightly dressed crowds in the background), there is always a trade-off between high video quality and low bitrate. Higher video quality requires higher bitrate. There is less trade-off with visually simple video such as cartoons.

AWS Elemental Live offers several options that provide different balances of video quality versus bitrate.

Do the following to set the rate control mode and bitrate for the output:

- In Event > Streams > Video > Video Code, choose H.264
- In Event > Streams > Video > Advanced > Rate Control Mode for Video, for Codec settings, select the desired option. For information about choosing the best option, see the sections below.
- Complete other fields, as appropriate:
  - If you select QVBR, complete Max bitrate, Quality level, Buffer size, and Buffer fill percentage.
  - If you select VBR, complete Bitrate (average bitrate), Max bitrate, Buffer size, and Buffer fill percentage.
  - If you select CBR, complete Bitrate, Buffer size, and Buffer fill percentage.

Note
We do not recommend use of either the CQ or ABR options. The Statmux option only applies if you have the Statmux option enabled on AWS Elemental Live and only if you are creating a "statmux output".

Quality-defined Variable Bitrate Mode (QVBR)

With quality-defined variable bit rate mode (QVBR), you specify a maximum bitrate and a quality level. Video quality will match the specified quality level except when it is constrained by the maximum bitrate. This constraint occurs when the video is very complex, so that it is not possible to reach the quality level without exceeding the maximum bitrate.

We recommend this mode if you or your viewers pay for bandwidth, for example, if you are delivering to a CDN or if your viewing users are on mobile networks.

When selecting QVBR, set the quality level and maximum bitrate for your most important viewing devices. Set the buffer size to twice the maximum bitrate, and set the initial buffer to 90%. Set the Buffer size to twice (2x) the Max bitrate and set the Buffer fill percentage to 90%. The buffer contributes to a smooth playout.

The following table lists the recommended values for this mode:

<table>
<thead>
<tr>
<th>Viewing Device</th>
<th>Quality Level</th>
<th>Max Bitrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary screen</td>
<td>8 to 10</td>
<td>4,000,000 to 6,000,000</td>
</tr>
<tr>
<td>PC or tablet</td>
<td>7</td>
<td>1,500,000 to 3,000,000</td>
</tr>
<tr>
<td>Smartphone</td>
<td>6</td>
<td>1,000,000 to 1,500,000</td>
</tr>
</tbody>
</table>

With this mode, the bitrate can change with each frame (in order to obtain at least the specified quality), but it cannot exceed the maximum bitrate. AWS Elemental Live does not attempt to maintain an average bitrate. It always hits the maximum bitrate if that is necessary to obtain the specified quality. On the other hand, if the quality can be obtained with lower bitrates, AWS Elemental Live does not use a higher bitrate.

Variable Bitrate Mode (VBR)

With variable bitrate mode (VBR), you specify an average bitrate and a maximum bitrate. Video quality and bitrate vary, depending on the video complexity.
Select VBR instead of QVBR if you want to maintain a specific average bitrate over the duration of the channel. If bitrate does not need to be constrained, then consider using QVBR.

When selecting VBR, try to assess the expected complexity of the video, and set a suitable average bitrate. Set the maximum bitrate to accommodate expected spikes. Set the Buffer size to twice (2x) the Max bitrate and set the Buffer fill percentage to 90%. The buffer contributes to a smooth playout.

With this mode, the bitrate can change with each frame (in order to obtain the best quality) but it cannot exceed the specified maximum bitrate. AWS Elemental Live also ensures that as the channel progresses, the stream meets the specified average bitrate. This mode is useful when you expect short spikes in the complexity of the video. AWS Elemental Live aims for the average bitrate but spikes to the maximum bitrate for a short time when necessary.

### Constant Bitrate Mode (CBR)

With constant bitrate mode (CBR), you specify a bitrate. Video quality varies, depending on the video complexity.

Select CBR only if you distribute your assets to devices that cannot handle variable bitrates. But if it is acceptable for the bitrate to occasionally differ from a specified rate, then consider using VBR or QVBR instead. Over the duration of the channel, you might obtain both a lower bitrate and better quality with VBR or QVBR.

When selecting CBR, set the bitrate to balance the video quality and the output bitrate. Set the Buffer size to twice (2x) the Max bitrate and set the Buffer fill percentage to 90%. The buffer contributes to a smooth playout.

With this mode, the output always matches the specified bitrate. Sometimes that bitrate results in higher-quality video, and sometimes it results in lower-quality video.

---

**Handling Video Quality**

**Topics**
- Image Processing Controls (p. 25)
- Encoding Controls (p. 33)

**Image Processing Controls**

**Topics**
- Image Processing – Scaling Content (p. 25)
- Image Processing – Color Correction (p. 26)
- Image Processing – Scan Type – Key Controls (p. 27)
- Image Processing – Scan Type – Secondary Fields (p. 30)
- Image Processing – Noise Reduction (p. 31)
- Image Processing – Framerate Conversion (p. 32)

**Image Processing – Scaling Content**

**Description**

Scaling content is typically applied to adjust source content to better match the intended output device screen size or to optimize the resolution of a video output for a particular bitrate. Typically, the lower the bitrate, the smaller the resolution. These settings relate to scaling.
Image Processing Controls

- **Resolution** (width, height): Controls the number of pixels encoded (e.g. 1280x720, 640x480)
- **Stretch to Output**: Controls the aspect ratio of video content itself. For example, if Stretch to Output is enabled and source content with a resolution of 1280x720 (16:9) is transcoded to 640x480 (4:3), the resulting output video appears skewed as the aspect ratio of the video content is modified.
- **Anti-alias**: Choose Bilinear Interpolation or the variable-tap Lanczos filter. In almost all cases, the Lanczos filter provides much better quality when scaling video content, with a small impact on performance.

**Recommendations**

Enable **Anti-alias** unless speed is of utmost importance and video quality is not.

**Location of Fields**

<table>
<thead>
<tr>
<th>Stream – Video &gt; Resolution</th>
<th>stream_assembly/video_description/resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream – Video &gt; Stretch to output</td>
<td>stream_assembly/video_description/stretch_to_output</td>
</tr>
<tr>
<td>Stream – Video &gt; Anti-alias</td>
<td>stream_assembly/video_description/anti_alias</td>
</tr>
</tbody>
</table>

Image Processing – Color Correction

**Description**

In some situations, color correction can be used to adjust the visual color representation of content:

- **Brightness / Contrast / Hue / Saturation**: Provide basic pixel luma and chroma adjustment.
- **Video Range**: Expands the pixel value range from 16-235 vs 0-255 (full-swing encoding). Set to true to set to full-swing encoding or set to false to pass through the encoding.
  
  Set to "true" only if you are sure the range of the input is not full-swing. If it is full-swing and you choose "true", the blacks and whites in the output are clipped.
- **Color Space Conversion**: Adjusts pixel values from input to output color space (e.g. REC-601).
  
  - Set to "None" to perform no conversion.
  - Set to "Force 601" or "Force 709" to convert inputs with differing colorspaces. When choosing either of these options, either the metadata must specify the colorspace of the input or you must manually specify that colorspace via the Color Space field in the Video Selector section. If the input colorspace is not specified in one of these ways, AWS Elemental Live ignores the selected value and does not do any conversion.

**Recommendations**

Color adjustment does not affect output quality so there are no specific recommendations.

**Location of Fields**

The following table lists each field by its location on the web interface and shows the location of the corresponding tag in the XML.

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input &gt; Video Selector &gt; Color Space</td>
<td>input/video_selector/color_space</td>
</tr>
</tbody>
</table>
Image Processing – Scan Type – Key Controls

Description

You can convert the scan type of the input to a different scan type: progressive, interlaced, hard telecine, or soft telecine. You can configure to leave the scan type as is or to convert from one incoming type (or a mix of incoming scan types) to another single type. Configuring for scan type conversion involves setting fields in specific ways. The three key fields to convert the scan type of the input are Configuration - Deinterlace Mode, Configuration - Interlace Mode, and Configuration - Telecine. The following table describes how to set these three key fields to convert a given input to a given output.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Configuration - Deinterlace Mode</th>
<th>Configuration - Interlace Mode</th>
<th>Configuration - Telecine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Progressive</td>
<td>Progressive</td>
<td>Off</td>
<td>Progressive</td>
<td>None</td>
</tr>
<tr>
<td>Interlaced</td>
<td>Progressive</td>
<td>Deinterlace</td>
<td>Progressive</td>
<td>None</td>
</tr>
<tr>
<td>Interlaced</td>
<td>Progressive</td>
<td>Adaptive</td>
<td>Progressive</td>
<td>None</td>
</tr>
<tr>
<td>Hard telecine</td>
<td>Progressive</td>
<td>Inverse telecine</td>
<td>Progressive</td>
<td>None</td>
</tr>
<tr>
<td>Hard telecine</td>
<td>Progressive</td>
<td>Adaptive</td>
<td>Progressive</td>
<td>None</td>
</tr>
<tr>
<td>Soft telecine</td>
<td>Progressive</td>
<td>Off</td>
<td>Progressive</td>
<td>None</td>
</tr>
</tbody>
</table>
## Image Processing Controls

### Deinterlace Mode

This field applies an initial conversion for certain from/to conversions (as shown in the table above).

- **Deinterlace**: Applies a deinterlace algorithm to content. If AWS Elemental Live detects that the source content is already progressive, no deinterlacing is applied.
- **Inverse Telecine**: Converts hard telecine 29.97i to progressive 23.976p.
- **Adaptive**: Analyzes source content to determine whether to apply the deinterlace or inverse telecine algorithm on source content.

### Interlace Mode

This field applies an initial conversion for certain from/to conversions (as shown in the table above).

- **Progressive**: Converts content to a higher framerate (e.g., 29.97 fps to 59.94 fps, 29.97 to 60, or 25 to 50).

---

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Configuration - Deinterlace Mode</th>
<th>Configuration - Interlace Mode</th>
<th>Configuration - Telecine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed</td>
<td>Progressive</td>
<td>Adaptive</td>
<td>Progressive</td>
<td>None</td>
</tr>
<tr>
<td>Progressive</td>
<td>Hard telecine</td>
<td>Off</td>
<td>One of the other options</td>
<td>Hard telecine</td>
</tr>
<tr>
<td>Hard telecine</td>
<td>Hard telecine</td>
<td>Off</td>
<td>One of the other options</td>
<td>None</td>
</tr>
<tr>
<td>Soft telecine</td>
<td>Hard telecine</td>
<td>Off</td>
<td>One of the other options</td>
<td>Hard telecine</td>
</tr>
<tr>
<td>Mixed</td>
<td>Hard telecine</td>
<td>Off</td>
<td>One of the other options</td>
<td>Hard telecine</td>
</tr>
<tr>
<td>Interlaced</td>
<td>Interlaced</td>
<td>Off</td>
<td>One of the other options</td>
<td>None</td>
</tr>
<tr>
<td>Mixed</td>
<td>Interlaced</td>
<td>Off</td>
<td>One of the other options</td>
<td>None</td>
</tr>
<tr>
<td>Progressive</td>
<td>Soft telecine</td>
<td>Off</td>
<td>One of the other options</td>
<td>Soft telecine</td>
</tr>
<tr>
<td>Hard telecine</td>
<td>Soft telecine</td>
<td>Inverse telecine</td>
<td>One of the other options</td>
<td>Soft telecine</td>
</tr>
<tr>
<td>Hard telecine</td>
<td>Soft telecine</td>
<td>Adaptive</td>
<td>One of the other options</td>
<td>Soft telecine</td>
</tr>
<tr>
<td>Soft telecine</td>
<td>Soft telecine</td>
<td>Off</td>
<td>One of the other options</td>
<td>Soft telecine</td>
</tr>
<tr>
<td>Mixed</td>
<td>Soft telecine</td>
<td>Adaptive</td>
<td>One of the other options</td>
<td>Soft telecine</td>
</tr>
</tbody>
</table>

* Deinterlace Mode is an image processing control. Interlace Mode and Telecine are encoding controls.

**Note: Converting the Scan Type to Progressive**

If content is not being converted to a higher framerate, the deinterlacer outputs one frame for every two fields in the source content (i.e., 1080i30 content is converted to 1080p30). If the framerate is being doubled (e.g. 29.97 fps to 59.94 fps, 29.97 to 60, or 25 to 50), the deinterlacer converts each field into a frame.
This field controls video field order and how the scan type is represented in the output bitstream.

- **Progressive**: Encodes output as "progressive."
- **Top Field First** or **Bottom Field First**: Forces field polarity (top or bottom first) to the specified value, reordering fields if the source has a different order and encodes the result as interlaced.
- **Follow (Default Top)** and **Follow (Default Bottom)**: Produces interlaced output, with the output having the same field polarity as the source. Therefore:
  - If the source is "interlaced", the output is interlaced with the same polarity as the source (it follows the source). The output could therefore be a mix of “top field first” and “bottom field first.”
  - If the source is "progressive", the output is interlaced with "top field first” or “bottom field first” polarity (depending on which Follow option you chose).

Note: If the output codec is Microsoft VC-1, then the interlace mode is always set to "progressive."

**Telecine**

This field appears for MPEG-4 AVC and MPEG-2 only if the Streams > Advanced > framerate field is set to 29.970.

- **Hard**: Produce 29.97i output from 23.976 input.
- **Soft**: Produce 23.976; the player converts this output to 29.97i.

**Recommendations**

- Converting to progressive output always improves output quality and should be enabled in any use case where progressive output is required or acceptable.
- Interlace coding is inherently less efficient than progressive coding so use interlace coding only for content that has already been interlaced.
- The choice of deinterlacing algorithm is very subjective. **Motion Adaptive Interpolation** is generally recommended as the sharper image quality tends to provide the best perceived quality across a broad set of users. Use the Force Mode option for deinterlacing only when the input is known to be interlaced content incorrectly flagged as "progressive."
- Use the Force Mode option for Inverse Hard Telecine when the input is known to consist entirely of Hard Telecine content.

**Location of Fields**

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams &gt; Advanced &gt; Telecine</td>
<td>stream_assembly/video_description/codec/telecine</td>
</tr>
<tr>
<td></td>
<td>where codec is one of the following:</td>
</tr>
<tr>
<td></td>
<td>• h264_settings</td>
</tr>
<tr>
<td></td>
<td>• mpeg2_settings</td>
</tr>
<tr>
<td></td>
<td>• h265_settings</td>
</tr>
<tr>
<td></td>
<td>• prores_settings</td>
</tr>
<tr>
<td>Streams &gt; Advanced &gt; Interlace Mode</td>
<td>stream_assembly/video_description/codec/interlace_mode</td>
</tr>
<tr>
<td></td>
<td>where codec is one of the following:</td>
</tr>
</tbody>
</table>
Image Processing – Scan Type – Secondary Fields

Description

The scan type of the input can be converted to a different scan type: progressive, interlaced, hard telecine, or soft telecine. You can configure to leave the scan type as is or to convert from one incoming type (or a mix of incoming scan types) to another single type. Configuring for scan type conversion involves setting fields in specific ways.

Force Deinterlace

The field applies only when the Deinterlacer Preprocessor is turned on. It deals with issues of badly tagged frames.

- When the Deinterlace Mode is Adaptive, set Force Mode to Off.
- When the Deinterlace Mode is Deinterlace, use Force Mode as follows:
  - Off: The processor does not convert frames that are tagged in metadata as "progressive." It only converts those that are tagged as some other type.
  - On: The processor converts every frame to "progressive" – even those that are already tagged as "progressive." Turn Force Mode on only if the input frames are incorrectly marked as "progressive." Do not turn otherwise; processing frames that are already tagged as "progressive" degrade video quality.
- When the Deinterlace Mode is Inverse Telecine, use Force Mode as follows:
  - Off: The processor monitors presence/absence of the hard telecine field repeat cadence and applies only to hard telecine to progressive conversion on frames that have a distinct cadence.
  - On: The processor still monitors for hard telecine cadence and adapts to cadence, but all frames are converted from hard telecine to progressive using the last detected cadence.

Deinterlace Algorithm

The field applies only when the Deinterlacer Preprocessor is turned on and when the Deinterlace Mode is set to Deinterlace or Adaptive. Set it to the desired value:

- Motion adaptive interpolation: Provides for better spatial quality (i.e. produces sharper images).
- Motion adaptive blend: Provides for better temporal quality (i.e. produces smoother motion).
• Low latency interpolation: Performs interpolated line-doubling to allow for lower latency applications.

Location of Fields

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams &gt; Advanced &gt;Preprocessors &gt; Deinterlacer &gt; Deinterlace Algorithm</td>
<td>stream_assembly/video_description/video_preprocessors/deinterlacer/algorithm</td>
</tr>
<tr>
<td>Streams &gt; Advanced &gt;Preprocessors &gt; Deinterlacer &gt; Force Mode</td>
<td>stream_assembly/video_description/video_preprocessors/deinterlacer/force</td>
</tr>
</tbody>
</table>

Image Processing – Noise Reduction

Description

In some applications, you can apply the Noise Reducer Preprocessor to improve output quality. It works by reducing spatial noise, which makes images compress better. But it changes the visual quality of the video.

• The Filter field has four options:
  • Mean / Gaussian / Lanczos: All of these algorithms allow for varying blur strengths. Mean is the strongest filter (it operates on a smaller group of pixels) while Lanczos is the mildest (it operates on a larger group of pixels).
  • Sharpen: This algorithm sharpens the edges instead of softening them.
  • Conserve: This algorithm limits the pixel values to within the minimum and maximum values of the neighboring pixel values. It is designed to reduce speckle noise. This filter does not seem to be very valuable with video.
  • Bilateral: This algorithm tends to preserve strong edges and is the best compromise between noise reduction and visual quality.
  • Strength: The strength field of the filtering has its greatest effect at 3.

Recommendations

In most cases, enabling the Noise Reducer is not required, but can help output quality if the content will be compressed heavily. When enabled, the recommended algorithm is Bilateral, but the other algorithms may produce better results in certain use cases. Testing the different algorithms using the expected source content is the recommended method to determine the best option.

Location of Fields

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams &gt; Advanced &gt;Preprocessors &gt; Noise Reducer &gt; Filter</td>
<td>stream_assembly/video_description/video_preprocessors/noise_reducer/filter</td>
</tr>
<tr>
<td>Streams &gt; Advanced &gt;Preprocessors &gt; Noise Reducer &gt; Strength</td>
<td>stream_assembly/video_description/video_preprocessors/noise_reducer/strength</td>
</tr>
</tbody>
</table>
Image Processing – Framerate Conversion

Description

framerate conversion is typically used when producing content for devices that use different standards (e.g. NTSC vs. PAL) or different content playback scenarios (e.g. film at 24 fps vs. television at 25 fps or 29.97 fps). There are a few different encoding settings that can be adjusted when performing framerate conversion:

- **framerate**: Defines the framerate of the output. The AWS Elemental system has some common built-in settings, but you can define custom settings.

- **Interpolated**: If Interpolated is disabled, the AWS Elemental engine drops or repeats frames as needed. This results in sharp individual frames. If Interpolated is enabled, a weighted average is applied between frames when “new” frames need to be added. This results in smoother motion. For example, when converting from a 24 fps input to 29.97 fps output, AWS Elemental Live uses an algorithm to average the 4th and 5th frames of the source content to produce the additional frame needed in the output.

- **Slow PAL**: This field appears only when framerate specifies 25 and applies when you are converting content from 23.976 fps to 25 fps frame rates. Slow PAL may be used to re-label the content as 25 fps and speed up the audio to compensate for the slower playback.

Recommendations

- If possible, try to avoid framerate conversion if you want to provide the best video quality. However, given workflow requirements or playback devices, it may not always be possible to avoid framerate conversion.

- Enable **Interpolation** if input and output frame rates are close (e.g. 24 fps inputs to 25 fps outputs).

Location of Fields

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream – Video &gt; Advanced &gt; Framerate</td>
<td>stream_assembly/video_description/codec/framerate_numerator</td>
</tr>
<tr>
<td></td>
<td>stream_assembly/video_description/codec/framerate_denominator</td>
</tr>
<tr>
<td>where codec is one of the following:</td>
<td></td>
</tr>
<tr>
<td>• h264_settings</td>
<td></td>
</tr>
<tr>
<td>• vc1_settings</td>
<td></td>
</tr>
<tr>
<td>• mpeg2_settings</td>
<td></td>
</tr>
<tr>
<td>• h265_settings</td>
<td></td>
</tr>
<tr>
<td>• prores_settings</td>
<td></td>
</tr>
</tbody>
</table>

| Stream – Video > Advanced > Interpolated | stream_assembly/video_description/codec/interpolate_frc |
| where codec is:                          | |
| • h264_settings                         | |
| • vc1_settings                          | |
Encoding Controls

Topics

- Encoding – Group of Pictures (GOP) (p. 33)
- Encoding – Rate Control (p. 36)
- Encoding – Rate Control Modes (p. 37)
- Encoding – Rate Control Tuning (p. 39)
- Encoding – Statmux Rate Control (p. 39)
- Encoding – Quantization Controls (p. 41)
- Encoding – Scan Type (p. 43)
- Encoding – MPEG-4 AVC (H.264) Controls (p. 43)
- Encoding – HEVC (H.265) Controls (p. 44)

Encoding – Group of Pictures (GOP)

Description

The Group of Pictures (GOP) settings define the basic pattern of the video stream itself in terms of how the encoding engine uses I-, P-, and B- frames. A few encoding settings control the GOP structure are as follows:

- **GOP Mode**: Select Fixed.
- **GOP Size**: Defines the interval between I-frames.
- **B-Frames**: Defines the maximum run of B-frames. **(Note: The encoding engine may make the decision to use a smaller run of B- frames in specific instances within the GOP if it determines this will produce higher quality content.**)
- **Closed GOP Cadence**: Defines the number of GOPs across which P- or B-frames are allowed to predict for encoding purposes.
- **Scene Change Detect**: Enables an algorithm that determines when a scene change occurs and inserts an I-frame.
- **Min I-interval**: Specifies a minimum number of frames between GOP Cadence I-frames and Scene Change Detect I-frames. I-frames require more bits than P- or B-frames, so encoding two in quick succession can hurt quality, particularly with small buffer sizes.
**GOP Reference B-Frame** (H.264 and H.265 only): Enables the use of reference B-frames for GOP structures that have B-frames greater than 1.

**Recommendations**

- **GOP Mode**: Always choose **Fixed**. The **Follow** mode is obsolete and not recommended.
- **GOP Size**: When using the MPEG-2 codec, the recommended **GOP Size** is up to 30 (15 is also very common). For H.264 or VC-1 codecs, our recommendation is to make this as large as possible while still meeting other encoding requirements.

  For example, for adaptive bitrate delivery in which a segment size of 6 seconds is used for 29.97 fps outputs, the largest GOP size should be 180 frames.

- **B Frames and GOP Reference B-Frames**: When using H.264 or H.265, enable **GOP Reference B-Frame** to obtain the best quality and set B-frames to a value from 3 to 5 (3 is recommended). For other codecs, there is no quality benefit to setting the B-frames to more than 2. For high-motion content, use 0 or 1 to produce the best quality.

- **Closed GOP Cadence**: For segmented outputs (e.g. for adaptive bitrate content, such as HLS, Smooth, HDS, DASH, etc.), set the **Closed GOP Cadence** to 1. For non-segmented outputs, this can be set to 0 to allow for an open GOP.

- **Scene Change Detect**: Always enable this for the best quality. The only scenario where this might be disabled is if a STB or playback device is unable to accommodate an additional I-frame within the normal GOP pattern. (This is rare.)

Some service providers, encoding vendors, or manufacturers may state that **Scene Change Detect** should be disabled during encoding of content. This recommendation is typically based on the fact that these providers have systems that require a consistent GOP structure and some encoders "reset" the GOP structure when an I-frame is inserted for a scene change.

With AWS Elemental Live, **Scene Change Detect** can be enabled in almost all cases as the default behavior is to simply insert an additional I-frame on a scene change but not disrupt the normal GOP structure defined in the encoding settings. The intent behind this approach is to ensure that adaptive bitrate outputs are always GOP-aligned, and, as a side effect, allow for compatibility with these third-party systems even with **Scene Change Detect** enabled.

- **Min I-interval** is enforced by shifting the GOP cadence; so, for segmented outputs that require a fixed cadence of I-frames, it cannot be used and should be set to 0. For non-segmented outputs, a **Min I-Interval** of 5 is recommended.

**Location of Fields**

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams – Video &gt; Advanced &gt; GOP Size</td>
<td>stream_assembly/video_description/codec/gop_size</td>
</tr>
<tr>
<td></td>
<td>where codec is one of the following:</td>
</tr>
<tr>
<td></td>
<td>• h264_settings</td>
</tr>
<tr>
<td></td>
<td>• vc1_settings</td>
</tr>
<tr>
<td></td>
<td>• mpeg2_settings</td>
</tr>
<tr>
<td></td>
<td>• h265_settings</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; B Frames</td>
<td>stream_assembly/video_description/codec/gop_num_b_frames</td>
</tr>
<tr>
<td>Location of Field on Web Interface</td>
<td>Location of Tag in XML</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Closed GOP Cadence</td>
<td>stream_assembly/video_description/codec/gop_closed_cadence</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Scene Change Detect</td>
<td>stream_assembly/video_description/codec/transition_detection</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Min I-interval</td>
<td>stream_assembly/video_description/codec/min_i_interval</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; GOP Reference B-Frame</td>
<td>stream_assembly/video_description/codec/gop_b_reference</td>
</tr>
</tbody>
</table>

where `codec` is one of the following:
- h264_settings
- vc1_settings
- mpeg2_settings
- h265_settings
Encoding – Rate Control

Description

For AWS Elemental Live encoding, a buffer model (such as VBV or HRD) is used to manage rate control. The buffer model is a conceptual model for specifying both short-term and long-term constraints on encoding stream bitrates. The concept is that bits flow into the buffer at a fixed rate and picture are extracted instantaneously. In a "compliant" stream, the buffer cannot overflow or underflow. The following are several settings that allow the user to adjust the rate control settings tied to this buffer model:

- **Bitrate**: Defines the long-term average bitrate, in bits/second. This field does not appear when Rate Control Mode is set to CQ.
- **Buffer Size**: Defines the total buffer size in bits.
- **Max Bitrate**: Defines the buffer fill rate, in bits/second. This field appears only when Rate Control Mode is set to VBR or Statmux.
- **Initial Buffer Fill**: Defines the percentage of the buffer filled with bits before the decode begins.

These fields work with the Rate Control Mode field, which is discussed in the section called “Encoding – Rate Control Modes” (p. 37).

Recommendations

- For **Bitrate**, more is better. The value used will depend more on other workflow constraints and will likely be different for each application.
- In general, **Buffer Size** is commonly set to 1x to 2x the bitrate (again, more is typically better).
- **Max Bitrate** is commonly set to 1x to 2x the bitrate. Although more is typically better, this value must also fit within the other encoding requirements of your specific application.
- For **Initial Buffer Fill**, the default value (90%) typically provides the best video quality, so leave it blank to use the default.

Location of Fields

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams – Video &gt; Advanced &gt; Bitrate</td>
<td>stream_assembly/video_description/codec/bitrate</td>
</tr>
<tr>
<td></td>
<td>where codec is one of the following:</td>
</tr>
<tr>
<td></td>
<td>• h264_settings</td>
</tr>
<tr>
<td></td>
<td>• vcl_settings</td>
</tr>
<tr>
<td></td>
<td>• mpeg2_settings</td>
</tr>
<tr>
<td></td>
<td>• h265_settings</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Buffer Size</td>
<td>stream_assembly/video_description/codec/buf_size</td>
</tr>
<tr>
<td></td>
<td>where codec is one of the following:</td>
</tr>
<tr>
<td></td>
<td>• h264_settings</td>
</tr>
<tr>
<td></td>
<td>• vcl_settings</td>
</tr>
</tbody>
</table>
Encoding Controls

Location of Field on Web Interface | Location of Tag in XML
--- | ---
Streams – Video > Advanced > Max Bitrate | stream_assembly/video_description/codec/max_bitrate

where codec is one of the following:
- h264_settings
- vc1_settings
- mpeg2_settings
- h265_settings

Streams – Video > Advanced > Initial Buffer Fill | stream_assembly/video_description/codec/buf_fill_pct

where codec is one of the following:
- h264_settings
- vc1_settings
- mpeg2_settings
- h265_settings

Encoding – Rate Control Modes

Description

Rate control modes define how the encoding engine uses the buffer model and how bits flow into the engine.

- The Rate Control Mode field has the following options.
  - **Constant Bitrate** (CBR) mode: Uses the buffer model described above where Max Bitrate equals the defined (Average) Bitrate. In this mode, AWS Elemental Live adds fill bits to the stream to ensure a constant bitrate video encoding.
  - **Variable Bitrate** (VBR) mode: Uses the buffer model described above where Max Bitrate is greater than or equal to the (Average) Bitrate. In this mode, AWS Elemental Live adds no fill bits for VBR video encoding.
  - **Average Bitrate** (ABR) mode: Does not use the buffer model but allows the bitrate to vary as needed while maintaining an overall average bitrate for the entire video encoding.
  - **Constant Quantizer** (CQ) mode: Disables rate control completely. AWS Elemental Live encodes frames using a constant **Quantization Parameter** (QP), which you can set using the Start QP setting.
  - **Statmux**: See the section called "Encoding – Statmux Rate Control" (p. 39).
  - **Start / Min / Max QP**: This field specifies the level of compression to start at when performing rate control and the minimum and maximum levels of compression.

Recommendations

- CBR mode provides the lowest quality at a given bitrate but has the benefit of providing minimal bitrate variation. It is supported by the widest variety of playback devices.
• **VBR** mode provides better quality than CBR but has more bitrate variation (though is still limited). VBR is supported by many modern devices but is not as widely supported as CBR, particularly with older playback devices.

• **ABR** mode provides the best quality but should only be used if bitrate variation is not critical.

• **CQ Mode** is a legacy control mode which does not typically provide any notable benefits for video quality tuning. Use of CQ mode is not recommended.

• **QP Mode** has an encoder that adjusts QP values optimally: leave all fields blank. Use of QP is not recommended.

### Location of Fields

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Streams – Video &gt; Advanced &gt; Rate Control Mode</strong></td>
<td><code>stream_assembly/video_description/codec/rate_control_mode</code> where <code>codec</code> is one of the following:</td>
</tr>
<tr>
<td></td>
<td>• <code>h264_settings</code></td>
</tr>
<tr>
<td></td>
<td>• <code>vc1_settings</code></td>
</tr>
<tr>
<td></td>
<td>• <code>mpeg2_settings</code></td>
</tr>
<tr>
<td></td>
<td>• <code>h265_settings</code></td>
</tr>
<tr>
<td><strong>Streams – Video &gt; Advanced &gt; Start QP</strong></td>
<td><code>stream_assembly/video_description/codec/qp</code> where <code>codec</code> is one of the following:</td>
</tr>
<tr>
<td></td>
<td>• <code>h264_settings</code></td>
</tr>
<tr>
<td></td>
<td>• <code>vc1_settings</code></td>
</tr>
<tr>
<td></td>
<td>• <code>mpeg2_settings</code></td>
</tr>
<tr>
<td><strong>Streams – Video &gt; Advanced &gt; Min QP</strong></td>
<td><code>stream_assembly/video_description/codec/min_qp</code> where <code>codec</code> is one of the following:</td>
</tr>
<tr>
<td></td>
<td>• <code>h264_settings</code></td>
</tr>
<tr>
<td></td>
<td>• <code>vc1_settings</code></td>
</tr>
<tr>
<td></td>
<td>• <code>mpeg2_settings</code></td>
</tr>
<tr>
<td><strong>Streams – Video &gt; Advanced &gt; Max QP</strong></td>
<td><code>stream_assembly/video_description/codec/max_qp</code> where <code>codec</code> is one of the following:</td>
</tr>
<tr>
<td></td>
<td>• <code>h265_settings</code></td>
</tr>
<tr>
<td></td>
<td>• <code>vc1_settings</code></td>
</tr>
<tr>
<td></td>
<td>• <code>mpeg2_settings</code></td>
</tr>
</tbody>
</table>
Encoding – Rate Control Tuning

Description

Following are encoding settings that can be used to provide additional tuning of video quality.

- **Lookahead**: This setting indicates that AWS Elemental Live should analyze a few frames in the future of the currently encoded frame (higher values mean more frames) and allow AWS Elemental Live to take future frame data into account during rate control logic.

  For example, if future frames are more complex, AWS Elemental Live can allocate fewer bits to encode the current frame to allow those bits to be used to encode those future frames. The tradeoff is that processing and latency are increased slightly to allow those future frames to be analyzed by the encoding engine.

Recommendations

- Use 2-pass encoding for VOD unless minimize encoding time is critical.
- Set **Lookahead** to "medium" for use with 1-pass encoding unless latency is critical.

Location of Fields

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams – Video &gt; Advanced &gt; Passes</td>
<td>stream_assembly/video_description/codec/passes</td>
</tr>
<tr>
<td></td>
<td>where codec is one of the following:</td>
</tr>
<tr>
<td></td>
<td>• h264_settings</td>
</tr>
<tr>
<td></td>
<td>• vc1_settings</td>
</tr>
<tr>
<td></td>
<td>• mpeg2_settings</td>
</tr>
<tr>
<td></td>
<td>• h265_settings</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Lookahead</td>
<td>stream_assembly/video_description/codec/lookAhead_rate_control</td>
</tr>
<tr>
<td></td>
<td>where codec is one of the following:</td>
</tr>
<tr>
<td></td>
<td>• h264_settings</td>
</tr>
<tr>
<td></td>
<td>• vc1_settings</td>
</tr>
<tr>
<td></td>
<td>• mpeg2_settings</td>
</tr>
<tr>
<td></td>
<td>• h265_settings</td>
</tr>
</tbody>
</table>

Encoding – Statmux Rate Control

Description

Statistical multiplexing (statmux) rate control applies only to an event created on an AWS Elemental Live node that has the Statmux option or a combination of an AWS Elemental Live node and a Statmux node.
The statmux controls are divided between the stream assembly of the event or profile and several fields on the MPTS output. The event is always created on the AWS Elemental Live node. The MPTS output is created on the AWS Elemental Live node or the Statmux node.

Statistical multiplexing is performed on the STPS channels that make up an MPTS output when the Rate Control Mode of the video stream is set to Statmux. In terms of video quality, all the controls described in the other sections of this chapter apply, along with the following special controls.

**SPTS Channel (Event) Controls**

In the event (which is called an SPTS channel in an MPTS output) the following controls apply:

- **Bitrate (Nominal Bitrate).**
- **Minimum Bitrate.** The bitrate allocated to a given SPTS channel, which will never go below this number.
- **Maximum Bitrate.** The bitrate allocated to a given SPTS channel, which will never exceed this number.

These settings guide the distribution of bitrate among the channels in the multiplex as described below.

**MPTS Output Controls**

The MPTS output has the following controls:

- **Transport Stream Bitrate:** The total size of the MPTS output.
- **Video Allocation:** The portion of the Transport Stream Bitrate that is allocated to the video of all the SPTS channels.

These settings determine the overall bitrate "pool" that is distributed among the channels in the multiplex as described below.

**Statmux Bitrate Allocation**

At runtime, each SPTS channel sends per-frame complexity information to the multiplexer and receives corresponding bitrate allocations. The allocations for each frame in channel are computed as follows.

- The **Nominal Bitrates** for all channels are scaled proportionately; such sum of the scaled nominal bitrates matches the Video Allocation for the multiplex. This is bitrate distribution if all the channels have the same complexity.
- The complexity of each channel is examined (relative to the other channels) and the **Nominal Bitrate** of each channel is adjusted. In this adjustment, the spread between the Min and Max of a given channel affects how much that channel's **Nominal Bitrate** is adjusted: the bigger the range, the bigger the adjustment.
- The **Nominal Bitrate** of each channel is adjusted again, if necessary, to stay within the Min and Max range.
- This last adjustment may result in "leftovers." These leftovers are distributed among channels that did not get adjusted.
- The final bitrate allocation is sent to each SPTS channel.

**Recommendations**

- **Transport Stream Bitrate:** The total size of the MPTS output; this number is set based on the bandwidth characteristics of the intended network.
- **Video Allocation:** This number is typically calculated as follows:

  \[ VA = \text{Useable bitrate} - (\text{the maximum anticipated audio of all SPTS channels}) - (\text{the maximum anticipated data of all SPTS channels}) \]
where the useable bitrate is typically 95-97% of the Transportation Stream Bitrate.

- **Nominal** bitrate is set for each SPTS channel relative to other channels depending on resolution, framerate, and codec used to encode the video. Typically the nominal bit rate is set to a value that provides the desired video quality in an non-statmuxed application.

- **Minimum** bitrate sets a minimum allocation for a channel regardless of how low its relative complexity. A typical value for each SPTS channel is:

\[(\text{Nominal}) \div 2\]

The combined minimum bitrates for all channels in the multiplex is less than the Video Allocation (VA). An alert is triggered if it ever equals or exceeds the VA.

- **Maximum** bitrate sets a maximum allocation for a channel, regardless of how high its relative complexity is. A typical value for each SPTS channel is:

\[\text{Nominal} \times 2\]

The combined maximum bitrate is typically greater than the combined Video Allocation (VA): if it is less than the VA, then all channels receive their maximum allocation (they are encoded at constant bitrates).

The allocation of bitrates in the MPTS can be monitored in real time via the MPTS Performance screen on the web interface.

**Location of Fields**

**Event or Profile API**

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams – Video &gt; Advanced &gt; Bitrate</td>
<td>stream_assembly/video_description/bitrate</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Min Bitrate</td>
<td>stream_assembly/video_description/min_bitrate</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Max Bitrate</td>
<td>stream_assembly/video_description/max_bitrate</td>
</tr>
</tbody>
</table>

**MPTS API**

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPTS &gt; Transportation Stream Bitrate</td>
<td>bitrate</td>
</tr>
<tr>
<td>MPTS &gt; Video Allocation</td>
<td>video_allocation</td>
</tr>
</tbody>
</table>

**Encoding – Quantization Controls**

**Description**

- **Adaptive Quantization (AQ):** Allows AWS Elemental Live to vary compression within a frame to improve subjective visual quality. It can distribute bits to provide more data to areas of a frame that are more complex for the encoding process. It can also reduce "smear" in high motion and sports content.
Note that although it may seem counter-intuitive, areas of solid color are considered more complex as human perception is more likely to notice small variations in smooth surfaces or gradients as opposed to more varied areas. For this reason, with **Adaptive Quantization**, more bits are allocated to smooth surfaces to minimize the perceptual variation between frames.

- **Framing Quantization**: An extension of **Adaptive Quantization** that compresses the edges of the video slightly more than the center. The effect shifts bits, and thus quality, from the boundary of the image to the middle of the image, where the action and viewers' attention is typically focused. This field appears only when the output codec is MPEG-2.
- **Softness**: Adjusts the quantization matrices used in the encoder, which determine the relative compression of high vs. low spatial frequency components of the video. Higher softness settings compress high frequencies more, reducing bitrate at the cost of image sharpness. This field appears only when the output codec is H.264 or MPEG-2.

### Recommendations

- **Adaptive Quantization**: For high bitrate outputs, we recommend that you always set **Adaptive Quantization** to “low.” For moderate bitrate outputs, our recommendation is "medium." For low bitrate outputs, our recommendation is "high" to "medium" or "high" for h.264 content. We generally recommend "high ", but, particular for low bitrate use cases, this setting can result in some use cases having too many bits be distributed to complex areas of the picture, resulting in more noticeable, lower quality in less complex areas. In those cases, "medium " may be more appropriate.

- **Framing Quantization**: For low bitrate encodes, we recommend using **Framing Quantization** (e.g. MPEG-2 1080i at 10 Mbps) and set based on subjective tuning at values between 1.0 and 2.0. (Note: The visual effect of framing quantization is intentionally subtle.)

- **Softness** For low bitrate encodes softness can be used (e.g. MPEG-2 1080i at 10 Mbps) and set based on subjective tuning at values between 24 and 32.

### Location of Fields

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams – Video &gt; Advanced &gt; Adaptive Quantization</td>
<td>stream_assembly/video_description/codec/adaptive_quantization</td>
</tr>
<tr>
<td><strong>Where codec</strong> is one of the following:</td>
<td></td>
</tr>
<tr>
<td>• h264_settings</td>
<td></td>
</tr>
<tr>
<td>• vc1_settings</td>
<td></td>
</tr>
<tr>
<td>• mpeg2_settings</td>
<td></td>
</tr>
<tr>
<td>• h265_settings</td>
<td></td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Framing Quantization</td>
<td>stream_assembly/video_description/codec/framing_quantization</td>
</tr>
<tr>
<td><strong>Where codec</strong> is:</td>
<td></td>
</tr>
<tr>
<td>• meg2_settings</td>
<td></td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Softness</td>
<td>stream_assembly/video_description/codec/softness</td>
</tr>
<tr>
<td><strong>Where codec</strong> is one of the following:</td>
<td></td>
</tr>
<tr>
<td>• h264_settings</td>
<td></td>
</tr>
</tbody>
</table>
Encoding – Scan Type

Description

The scan type of content can affect the video quality. The following are the settings and internal algorithms tied to the scan type:

- The encoding controls that deal with scan type are dealt with in the section called “Image Processing – Scan Type – Key Controls” (p. 27).
- **Picture Adaptive Field Frame (PAFF):** This control is automatically enabled on Graphics Processing Unit (GPU)-enabled versions of AWS Elemental Live and automatically disabled on Central Processing Unit (CPU)-only versions.
- **Macroblock Adaptive Field Frame (MBAFF):** This control is automatically enabled on CPU-only versions of AWS Elemental Live and automatically disabled on GPU-enabled versions.
- **Force Field Pictures:** This field appears only if the codec is H.264 and only affects GPU-enabled versions of AWS Elemental Live.
  - **Enabled:** All outputs are forced to use PAFF field picture encoding.
  - **Disabled:** AWS Elemental Live switches between PAFF and MBAFF, depending on the content.

Recommendations

- **Force Field Pictures** results in a significant reduction in quality so it should only be used if required for compatibility with specific decoders or playback devices.

Location of Fields

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams – Video &gt; Advanced &gt; Force Field Pictures</td>
<td>stream_assembly/video_description/codec/force_field_pictures</td>
</tr>
<tr>
<td></td>
<td>where codec is: h264_settings</td>
</tr>
</tbody>
</table>

Encoding – MPEG-4 AVC (H.264) Controls

Description

A few H.264-specific controls are available on AWS Elemental Live that affect video quality:

- **CABAC:** This control enables arithmetic coding and can compress the same data with ~15% fewer bits.
- **Slices:** This control improves speed of encoding. Using a higher number of slices can improve speed optimization but results in slightly less quality.
- **Flicker Reduction:** This control reduces flicker in the video. It enables an additional encoding algorithm that compensates for visual differences when I-frames have occurred (often referred to as I-
frame "pop"). AWS Elemental Live uses this approach to encode I-frames as if they were P-frames (thus creating a "PI"-frame), identify macroblocks that are prone to flicker, merge pixels from the PI-frame for the flicker-prone MBs into the I-frame, and then encode the PI-frame. This method carries some of the visual quality from the end of one GOP to the start of the next GOP, preventing the abrupt change in detail that gives the effect of a flicker in the video.

- **Reference Frames**: This control defines the number of frames that can be used to define B- and P-frames.

**Recommendations**

- Use **High Profile** with H.264 to improve quality over Baseline Profile and Main Profile.
- Always enable **CABAC** unless the intended decoder or playback device doesn't support it.
- Set **Slices** to 2 (or higher) for all HD resolution outputs or high bitrate outputs. (AWS Elemental Live presets typically use 2 slices for 720p content and 4 slices for 1080p content.) For resolutions below 720p, set **Slices** to 1.
- Set **Flicker Reduction** to "high" for low-motion content. Disable it for high-motion content.
- Set **Reference Frames** to 2 or more for flash compensation.

**Location of Fields**

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams – Video &gt; Advanced &gt; Profile</td>
<td>stream_assembly/video_description/h264_settings/cabac</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Level</td>
<td>stream_assembly/video_description/h264_settings/level</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; CABAC</td>
<td>stream_assembly/video_description/h264_settings/cabac</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Slices</td>
<td>stream_assembly/video_description/h264_settings/slices</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Flicker Reduction</td>
<td>stream_assembly/video_description/h264_settings/flicker_reduction</td>
</tr>
</tbody>
</table>

**Encoding – HEVC (H.265) Controls**

**Description**

The following HEVC-specific control can affect video quality:

- **Slices**: This control improves speed of encoding. Using a higher number of slices can improve speed optimization but results in slightly less quality.

**Recommendations**

The AWS Elemental Live HEVC encoder performance is less sensitive to changes in slices than the MPEG-4 AVC encoder (the section called "Encoding – MPEG-4 AVC (H.264) Controls" (p. 43)), so the benefit of using more slices is reduced and having more slices reduces video quality. For these reasons, the recommendation is to use half as many slices with HEVC as with MPEG-4 AVC for the same
resolution. In other words, set **Slices** to 2 (or higher) for all 1080p (or above) resolution outputs or high bitrate outputs. Set Slices to 1 for Resolutions below 1080p.

### Location of Fields

<table>
<thead>
<tr>
<th>Location of Field on Web Interface</th>
<th>Location of Tag in XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams – Video &gt; Advanced &gt; Profile</td>
<td>stream_assembly/video_description/h265_settings/cabac</td>
</tr>
<tr>
<td>Streams – Video &gt; Advanced &gt; Slices</td>
<td>stream_assembly/video_description/h265_settings/slices</td>
</tr>
</tbody>
</table>
Working with Audio

This chapter describes how to set up the audio features of AWS Elemental Live.

Topics
  • Working with Dolby Metadata (p. 46)
  • Setting up HLS Rendition Groups (p. 57)

Working with Dolby Metadata

Audio encoded with a Dolby codec always includes Dolby metadata, as per the ATSC A/52 2012 standard. This Dolby metadata is used by AWS Elemental Live in two ways when the stream is encoded with Dolby codec:

• It is used to manipulate the audio just before encoding the output.
• It is included in the metadata for the output stream.

This document describes how to set up an AWS Elemental Live profile or event to use Dolby metadata in these ways.

Dolby metadata is supported in the output only when the audio codec for the output is Dolby Digital (also known as AC3) or Dolby Digital Plus (also known as Enhanced AC3).

Topics
  • Categories of Metadata: Delivered and Encoder Control (p. 46)
  • Source of AWS Elemental Live Metadata (p. 47)
  • Impact of the Metadata on the Output Audio (p. 47)
  • Combinations of Input and Output Codec (p. 47)
  • Setting Up the Profile or Event Using the Web Interface (p. 48)
  • Output with the Dolby Digital Codec (p. 51)
  • Output with Dolby Digital Plus (EC2, EAC3) Codec (p. 54)

Categories of Metadata: Delivered and Encoder Control

There are two categories of parameters in the Dolby metadata, characterized by how AWS Elemental Live uses it:

• Delivered: AWS Elemental Live does not read these parameters, so they have no effect on the audio produced by AWS Elemental Live. Instead, they are included as metadata in the output in order to deliver them to the downstream decoder.

"Delivered" metadata is also called Consumer metadata because it is intended to be used by the end consumer's home decoder.
• Encoder Control: AWS Elemental Live uses these parameters to manipulate the audio just before encoding the stream and producing the output. They provide a mechanism for AWS Elemental Live to control the transcoding performed by AWS Elemental Live. These parameters are never included in the output metadata.

“Encoder Control” metadata is also called Professional metadata because it is intended to be used by a professional device – in our case AWS Elemental Live. It is never intended for the end consumer’s home decoder.

Source of AWS Elemental Live Metadata

The metadata that AWS Elemental Live emits can come from one of two sources:

• Metadata that is already in the source. Only audio sources that use a Dolby codec can include this metadata. Different Dolby codecs include different categories of metadata as shown in this table.

<table>
<thead>
<tr>
<th>Codec</th>
<th>Categories Present</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dolby Digital or Dolby Digital Plus</td>
<td>Delivered only</td>
</tr>
<tr>
<td>Dolby E</td>
<td>Delivered and Encoder Control</td>
</tr>
</tbody>
</table>

• Metadata that is specified by completing metadata fields in the profile or event. You can specify this metadata in any audio whose output codec is a Dolby codec. In other words, you can add it when the audio source is not a Dolby codec as long as the output audio uses a Dolby codec.

Both categories of metadata can be specified when specifying this source.

You specify the source when setting up the profile or event.

Impact of the Metadata on the Output Audio

Regardless of the source of the metadata, it affects the audio (either by manipulating encoder control or by being included in the output metadata) but only if the output codec is Dolby Digital or Dolby Digital Plus.

Combinations of Input and Output Codec

The possible input and output codec combinations (in which at least one codec is a Dolby codec) are as follows. All these combinations support including metadata in the output.

<table>
<thead>
<tr>
<th>Input Codec</th>
<th>Output Codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dolby Digital or Dolby Digital Plus</td>
<td>Dolby Digital or Dolby Digital Plus</td>
</tr>
<tr>
<td>Dolby Digital</td>
<td>Dolby Digital Passthrough (so Dolby Digital audio is passed through; it is not transcoded)</td>
</tr>
<tr>
<td>Dolby Digital Plus</td>
<td>Dolby Digital Passthrough (so Dolby Digital Plus audio is passed through; it is not transcoded)</td>
</tr>
<tr>
<td>Mix of Dolby Digital Plus and another codec</td>
<td>Dolby Digital Plus (with the Automatic Passthrough field checked)</td>
</tr>
<tr>
<td>Input Codec</td>
<td>Output Codec</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Dolby E</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td>Dolby E</td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td>Dolby E</td>
<td>Dolby E (passthrough)</td>
</tr>
<tr>
<td>A non-Dolby codec</td>
<td>Dolby Digital or Dolby Digital Plus</td>
</tr>
</tbody>
</table>

The sample rate when encoding with a Dolby codec is always 48000.

### Setting Up the Profile or Event Using the Web Interface

This section describes how to set up the project or event using the web interface. To set up using the REST API, see the section called “Output with AC3” (p. 51) and the section called “Output with EAC3” (p. 54) to map the fields to their XML tags according to the following steps.

1. In the Output > Stream section, click the Audio tab to display the fields for audio.
2. Choose one of these Audio Sources as appropriate and complete the Audio Codec field: Dolby Digital, Dolby Digital Plus or Dolby Digital Pass Through. The fields for metadata appear.
3. Complete the remaining fields for the audio source that you selected. See the following to determine how to achieve the desired effect.

**Dolby Digital**

For Dolby Digital, encoder control fields are circled in blue and delivery fields are circled in red. Note that the LFE Filter field appears only when the Coding Mode is 3/2 mode.

**Dolby Digital Plus**

Encoder Control fields are circled in blue. Delivery fields are circled in red. Note that the Automatic Pass-through field does not relate to metadata.

Note that the Surround Mode field appears only when Coding Mode is 2/0.
Dolby Digital Passthrough

There are no fields for metadata.

Use the Metadata in the Audio Source – Case 1

<table>
<thead>
<tr>
<th>Input Codec</th>
<th>Output Codec</th>
<th>Handling of Audio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dolby Digital or Dolby Digital Plus</td>
<td>Dolby Digital or Dolby Digital Plus</td>
<td>You are transcoding the audio.</td>
</tr>
</tbody>
</table>

- **Metadata Parameters** fields: Complete only AWS Elemental Live Control fields, as required for your workflow.
- **Follow Input Metadata** field: Check this field after completing the Metadata Parameter fields.

Result for Metadata

AWS Elemental Live Control parameters from the profile are applied during transcoding (given that the input does not include these parameters). If a given parameter is not exposed in the profile, a default value is always applied; see the section called “Output with AC3” (p. 51) and the section called “Output with EAC3” (p. 54).

The Delivery parameters from the input metadata are included in the output.

Use the Metadata in the Audio Source – Case 2

<table>
<thead>
<tr>
<th>Input Codec</th>
<th>Output Codec</th>
<th>Handling of Audio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dolby Digital or Dolby Digital Plus</td>
<td>Dolby Digital or Dolby Digital Plus (Passthrough)</td>
<td>You are passing through the audio.</td>
</tr>
</tbody>
</table>

- **Metadata Parameters** fields: Not applicable.
• **Follow Input Metadata** field: No Encoder Control parameters are applied (because no transcoding occurs).

**Result for Metadata**

The Delivery parameters from the input metadata will be included in the output.

**Use the Metadata in the Audio Source – Case 3**

<table>
<thead>
<tr>
<th>Input Codec</th>
<th>Output Codec</th>
<th>Handling of Audio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mix of Dolby Digital Plus and another codec</td>
<td>Dolby Digital Plus</td>
<td>You are passing through the Dolby Digital Plus audio and transcoding the remainder (Automatic Passthrough field is checked).</td>
</tr>
</tbody>
</table>

• **Metadata Parameters** field: Complete all the parameters.

• **Follow Input Metadata** field: Check this field after completing the metadata fields.

**Result for Metadata**

• AWS Elemental Live Control parameters from the profile will be applied when transcoding the non-Dolby Digital Plus audio.

• No Encoder Control parameters will be applied when passing through the Dolby Digital Plus audio.

• The Delivery parameters from the profile will be used when transcoding the non-Dolby Digital Plus audio.

• The Delivery parameters from the audio source will be used for the passed-through Dolby Digital audio.

**Use the Metadata in the Audio Source – Case 4**

<table>
<thead>
<tr>
<th>Input Codec</th>
<th>Output Codec</th>
<th>Handling of Audio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dolby E</td>
<td>Dolby Digital or Dolby Digital Plus</td>
<td>You are transcoding the audio.</td>
</tr>
</tbody>
</table>

• **Metadata Parameters** fields: Ignore.

• **Follow Input Metadata** field: Check.

**Result for Metadata**

• AWS Elemental Live Control parameters from the input metadata are applied during transcoding.

• The Delivery parameters from the input metadata are included in the output.
Override the Metadata with New Values – Case 5

<table>
<thead>
<tr>
<th>Input Codec</th>
<th>Output Codec</th>
<th>Desired Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any codec</td>
<td>Dolby Digital or Dolby Digital Plus</td>
<td>To override the metadata in the audio source.</td>
</tr>
</tbody>
</table>

- **Metadata Parameters** field: Complete as desired.
- **Follow Input Metadata** field: Leave unchecked.

**Result for Metadata**

The values from the profile are used.

- With all parameters except Dialnorm, the values from the profile are used. If a given parameter is not exposed in the profile, a default value is always applied; see the section called “Output with AC3” (p. 51) and the section called “Output with EAC3” (p. 54).
- With Dialnorm, the value from the profile is used. If the profile has no value and the source is a Dolby file, the value from the input metadata is used. If the profile has no value and the source is not a Dolby file, a default value is used; see the section called “Output with AC3” (p. 51) and the section called “Output with EAC3” (p. 54).

AWS Elemental Live Control parameters are applied during transcoding. The Delivery parameters are included in the output.

**Output with the Dolby Digital Codec**

<table>
<thead>
<tr>
<th>Nombre</th>
<th>Category</th>
<th>Value</th>
<th>Field</th>
<th>API Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calera</td>
<td>Dialogue</td>
<td>Level</td>
<td>Dialnorm</td>
<td>dialnorm</td>
</tr>
<tr>
<td>Calera</td>
<td>Channel</td>
<td>Mode</td>
<td>Coding</td>
<td>coding_mode</td>
</tr>
<tr>
<td>LFE</td>
<td>Channel</td>
<td>Mode</td>
<td>Coding</td>
<td>Coding_disabled_mode</td>
</tr>
</tbody>
</table>

Version 2.17

51
<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitstream Mode</td>
<td>Complete</td>
<td>Main</td>
</tr>
<tr>
<td>LinB Mode</td>
<td>No</td>
<td>Film</td>
</tr>
<tr>
<td>RF D Mode</td>
<td>No</td>
<td>Film</td>
</tr>
<tr>
<td>Overmodulation Protection Mode</td>
<td>No</td>
<td>user control</td>
</tr>
<tr>
<td>Center Downmix Level</td>
<td>-3dB</td>
<td>user control</td>
</tr>
<tr>
<td>Surround Downmix Level</td>
<td>Not indicated</td>
<td>user control</td>
</tr>
<tr>
<td>Dolby Mode</td>
<td>Disabled</td>
<td>user control</td>
</tr>
<tr>
<td>Audio Production Information</td>
<td>0</td>
<td>(does not exist)</td>
</tr>
<tr>
<td>Mix D Level</td>
<td>No</td>
<td>user set control</td>
</tr>
<tr>
<td>Room Type</td>
<td>No</td>
<td>user set control</td>
</tr>
<tr>
<td>Copyright Bit</td>
<td>0</td>
<td>user control</td>
</tr>
<tr>
<td>Switch</td>
<td>Caption</td>
<td>Name</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Original</td>
<td>Bitstream</td>
<td>control</td>
</tr>
<tr>
<td>Preferred</td>
<td>Stereomix Downmix</td>
<td>control</td>
</tr>
<tr>
<td>Lt/D</td>
<td>Downmix Level</td>
<td>0</td>
</tr>
<tr>
<td>Lt/D</td>
<td>Downmix Level</td>
<td>0</td>
</tr>
<tr>
<td>Lt/Rt</td>
<td>Downmix Level</td>
<td>0</td>
</tr>
<tr>
<td>Lo/Lo</td>
<td>Downmix Level</td>
<td>0</td>
</tr>
<tr>
<td>Dolby Surround</td>
<td>EX Mode</td>
<td>0</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>Type</td>
<td>0</td>
</tr>
<tr>
<td>DC Filter</td>
<td>Enabled</td>
<td>0</td>
</tr>
</tbody>
</table>
### Output with EAC3

<table>
<thead>
<tr>
<th>Category</th>
<th>Field</th>
<th>API</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Named Metadata</td>
<td>Parameters</td>
<td>stream_assembly</td>
<td>audio_description</td>
</tr>
<tr>
<td>EAC3 Settings</td>
<td>LFE Filter</td>
<td>lfe_filter</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

When Coding Mode is 3/2, LFE Filter checkbox appears at the far right.

<table>
<thead>
<tr>
<th>Surround Mode</th>
<th>3 dB Attenuation</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 dB control</td>
<td>No user control</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Surround Mode</th>
<th>Phase Shift</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>No user control</td>
<td>Disabled</td>
<td></td>
</tr>
</tbody>
</table>

### Output with Dolby Digital Plus (EC2, EAC3) Codec

<table>
<thead>
<tr>
<th>Category</th>
<th>Field</th>
<th>API</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Named Metadata</td>
<td>Parameters</td>
<td>stream_assembly</td>
<td>audio_description</td>
</tr>
<tr>
<td>Dialogue Level</td>
<td>Dialnorm</td>
<td>dials</td>
<td>Not set</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Channel Mode</th>
<th>Coding Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/2 – L,R,C,LS,Rs</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LFE Channel</th>
<th>LFE Filter</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When Coding Mode is 7 or 3/2, the LFE

---

Version 2.17
54
<table>
<thead>
<tr>
<th>Named Metadata Parameters</th>
<th>Category</th>
<th>Field</th>
<th>API</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream_assembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>audio_description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ac3_settings</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bitstream Mode
- Mode: Complete
- Mode: Main

LinB DRC
- Mode: Line
- Std. Compression: Main

RF D DRC
- Mode: RF
- Std. Compression: Main

RF D No-Overmodulation Protection

Center Downmix Level
- No: -3dB

Surround Downmix Level
- No: Not indicated

Dolby Surround Mode
- No: Disabled

Audio Production Information
- No: (does not exist)

Mix Level
- No: Not user set

Room Type
- No: User set control

Checkbox appears three lines below that checkbox.
### Output with EAC3

<table>
<thead>
<tr>
<th>Named Metadata Parameters</th>
<th>Category</th>
<th>Field</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copyright Bit</td>
<td>D</td>
<td>No</td>
<td>user control</td>
</tr>
<tr>
<td>Original Bitstream Bit</td>
<td>D</td>
<td>No</td>
<td>user control</td>
</tr>
<tr>
<td>Preferred Stereo Downmix</td>
<td>Not</td>
<td>Not indicated</td>
<td></td>
</tr>
<tr>
<td>Lt/Rt Center Mix Level</td>
<td>Lt/Lt</td>
<td>-3.0 dB</td>
<td></td>
</tr>
<tr>
<td>Rt/Rt Center Mix Level</td>
<td>Rt/Rt</td>
<td>-3.0 dB</td>
<td></td>
</tr>
<tr>
<td>Lt/Rt Surround Mix Level</td>
<td>Lt/Lt</td>
<td>-3.0 dB</td>
<td></td>
</tr>
<tr>
<td>Rt/Rt Surround Mix Level</td>
<td>Rt/Rt</td>
<td>-3.0 dB</td>
<td></td>
</tr>
<tr>
<td>Lo/Ro Center Mix Level</td>
<td>Lo/Lo</td>
<td>-3.0 dB</td>
<td></td>
</tr>
<tr>
<td>Ro/Ro Center Mix Level</td>
<td>Ro/Ro</td>
<td>-3.0 dB</td>
<td></td>
</tr>
<tr>
<td>Dolby Surround EX Mode</td>
<td>Disabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A/D Converter Type</td>
<td>0</td>
<td>No user control</td>
<td></td>
</tr>
<tr>
<td>DCEC DC dc Filter</td>
<td>Enabled</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## HLS Rendition Groups

In AWS Elemental Live, you can set up an HLS output group to support an audio rendition group.

In setting up an HLS output group to support an audio rendition group, each HLS output you create contains a "set" consisting of one video stream and several audio streams. All the audio streams in the set are associated with that one video stream. The HLS output group can contain more than one of these outputs. For example, one set consisting of high bitrate video and audio in four languages and another set consisting of low bitrate video and audio in the same four languages.

With this setup, the manifest that is created provides options for video. The logic of the manifest allows the player to select one of those video options and then to select audio that is valid for that video option.

For example:

1. The client player reads the manifest and selects the desired video, such as a high bitrate video.
2. The client player then selects an audio group from among the groups associated with that video, such as the Dolby Digital group instead of the AAC group.
3. The client player then selects an audio from that group, such as Spanish.

Typically, the player makes its audio selection based on rules on the player side, such as selecting the language that corresponds to the operating system language, or based on rules defined in the manifest, such as when the manifest identifies one audio as the default.
Standards Compliance

This implementation of audio rendition groups is compliant with the “HTTP Live Streaming draft-pantos-http-live-streaming-18” section 4.3.4.1.1.

Note that AWS Elemental Live does not support rendition groups for video. They do support rendition groups for captions since AWS Elemental Live automatically creates one captions rendition group to hold all caption stream assemblies in a given output.

Topics

• How Video Is Associated with Audio Rendition Groups (p. 58)
• Rules for Rendition Groups (p. 58)
• Examples of HLS Rendition Groups (p. 59)
• Creating HLS Rendition Groups (p. 61)
• Sample HLS Output Group with Audio Rendition Group Event Manifest (p. 73)

How Video Is Associated with Audio Rendition Groups

The different “sets” of media are created as follows:

• Create one “video-only” stream assembly containing only one video stream.
• Create two or more “audio-only” stream assemblies, each containing only one audio stream.
• Assign “audio group IDs” to the audio-only streams. To group several audio streams into one rendition group, assign the same audio group ID to the relevant audio streams. To group other audio streams into another rendition group, define a different audio group ID and assign it to the relevant audio streams.
• Associate each video-only stream with its corresponding audio rendition group by assigning the desired audio group ID to that stream.

For example:

• To group stream 3, 4 and 5 to one audio rendition group, set the audio group ID for each of these streams to “audio 1” or some other name of your choosing.
• To group streams 6, 7 and 8 to another audio rendition group, set the ID for each of these streams to "audio 2" or some other name.
• To associate video 1 with the first rendition group, set the “audio rendition sets ID” of that video to “audio 1”.
• To associate video 2 with the other group, set the audio rendition sets ID to “audio 2.”

Rules for Rendition Groups

Rules exist for associating both audio and video streams in their respective rendition groups. These are described following.

• A given audio stream can belong to only one audio rendition group.
• Any given video stream can be associated with more than one rendition group. (For example, “video high” can be associated with both “Dolby audio streams” and “AAC audio streams.” You do not need to create two video streams.)

With this setup, all the rendition groups associated with the same video stream must contain the same audio streams. (For example, “Dolby audio streams” and “AAC audio streams” must contain the same audio streams (perhaps English, French and Spanish)).
• Any given audio rendition group can be associated with more than one video stream. (For example, “Dolby audio streams” rendition group can be associated with “video high” and “video low.” You do not need to create two rendition groups, one for each video.)

• Any video stream can be associated with more than one output group. (For example, “video high” can appear in two different HLS output groups).

You can use a combination of these rules. For more information, see Examples of HLS Rendition Groups (p. 59).

Examples of HLS Rendition Groups

Example 1

The outputs in an HLS output group consist of:

• One video stream.

This video is associated with an audio rendition group that contains:

• One English stream.
• One French stream.
• One Spanish stream.

Example 2

The outputs in an HLS output group consist of:

• One “video high” stream.
• One “video medium” stream.
• One “video low” stream.

Each of these videos is associated with the same audio rendition group that contains:

• One English stream.
• One French stream.
• One Spanish stream.
Example 3

The outputs in an HLS output group consist of:

- One video stream.

This video is associated with two audio rendition groups.

The first audio rendition group contains:

- One English stream in AAC codec.
- One French stream in AAC codec.
- One Spanish stream in AAC codec.

The second audio rendition group contains:

- One English stream in Dolby Digital.
- One French stream in Dolby Digital.
- One Spanish stream in Dolby Digital.

Example 4

There are two output groups, one that pushes to a WebDAV server and the other that delivers to an Akamai server.

Each output group is identical in terms of its video and rendition groups. For example, each output group produces the video and rendition group from Example 2. You do not need to encode the streams twice;
do it only once for each output group. So long as the two output groups are in the same event, each can be associated with the same streams.

Example 5

There are two output groups, one that pushes to a WebDAV server and the other that delivers to an Akamai server.

Each output group is similar in terms of its video and rendition groups. For example, the first output group produces the video and rendition group from Example 2. The second output group produces the only “video high” and “video low” but it is associated with the same audio rendition group as the first output group.

Creating HLS Rendition Groups

The key to creating rendition groups is that each output you create must contain only one stream. Therefore, for each video to include in the output group, you create a stream assembly that contains only one video (no audio or captions). For each audio to include in a rendition group, you create a stream assembly that contains only one audio (no video or captions).

This means that when rendition groups are present in the HLS output group, an output is identical to a stream. (Usually an output contains a mix of several streams and several stream types.)

Topics
- Getting Ready to Create HLS Rendition Groups (p. 62)
- Creating HLS Rendition Groups (Web Interface) (p. 64)
- Creating HLS Rendition Groups (REST API) (p. 67)
Getting Ready to Create HLS Rendition Groups

Step 1. Create a Mapping

Identify the video, audio, audio rendition groups and captions you require. Review the section called “Rules for Rendition Groups” (p. 58) to ensure you design an output that is valid. For example:

- Video “high definition.”
- Video “low definition.”
- A rendition group named “AAC group” for AAC audio.
- A rendition group named “Dolby group” for Dolby Digital audio.
- Audio English AAC in “AAC group” rendition group.
- Audio English Dolby Digital in “Dolby group” rendition group.
- Audio French AAC in “AAC group” rendition group.
- Audio French Dolby Digital in “Dolby group” rendition group.
- Video “high definition” to be associated with both rendition groups.
- Video “low definition” to be associated with the “Dolby group” rendition group.
- Captions in English and French in WebVTT format, to be associated with both rendition groups.

![Diagram of output group, video, audio streams, and captions]

Step 2. Determine Defaults and Auto-Selection Behavior

For each audio rendition group, decide which audio will be the default and how auto-selection works for the non-defaults. Setting up this information is useful if:

- The user has specified an audio preference on the client player but that preference is not available, or
- If the user has not specified an audio preference.

(Obviously, if the user has specified a preference and that preference is available, the client player will select that preference.)

Determine Defaults and Auto-Selection Behavior

Set the Audio Track Type field. The options for this field for each audio stream follow.
<table>
<thead>
<tr>
<th>Value</th>
<th>Client Player Behavior</th>
<th>Representation in HLS Manifest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate Audio, Auto Select, Default</td>
<td>The client player should <em>select</em> this stream. Only one stream in the rendition group should be set as the default; otherwise, the client player may behave unexpectedly.</td>
<td>EXT-X-MEDIA with DEFAULT=YES, AUTOSELECT=YES</td>
</tr>
<tr>
<td>Alternate Audio, Auto Select, Not Default</td>
<td>The client player <em>may</em> select this stream. Any number of streams in the rendition group can be set this way.</td>
<td>EXT-X-MEDIA with DEFAULT=NO, AUTOSELECT=YES</td>
</tr>
<tr>
<td>Alternate Audio, not Auto Select</td>
<td>The client player <em>should never</em> select this stream. Any number of streams in the rendition group can be set this way.</td>
<td>EXT-X-MEDIA with DEFAULT=NO, AUTOSELECT=NO</td>
</tr>
<tr>
<td>Audio-Only Variant Stream</td>
<td>The client can play back this audio-only stream instead of video in low-bandwidth scenarios.</td>
<td>EXT-X-STREAM-INF</td>
</tr>
</tbody>
</table>

1. Set the **Alternate Audio Track Selection** as follows:

<table>
<thead>
<tr>
<th>Desired Result</th>
<th>How to Set</th>
</tr>
</thead>
</table>
| There is a default. The player can auto-select any of the other audios. | • Set only one audio stream to “Alternate Audio, Auto-Select, Not Default, Default.”  
  • Set every other audio stream to “Alternate Audio, Auto-Select, Not Default.” |
| There is a default. The player cannot auto-select any of the other audios. | • Set only one audio stream to “Alternate Audio, Auto-Select, Not Default, Default.”  
  • Set every other audio stream to “Alternate Audio, not Auto-Select.” |
| There is a default. There are specific audios that the player can auto-select. | • Set only one audio stream to “Alternate Audio, Auto-Select, Not Default, Default.”  
  • Set some of the other audio streams to “Alternate Audio, Auto-Select, Not Default.”  
  • Set some of the other audio streams to “Alternate Audio, not Auto-Select.” |
| There is no default. The player can auto-select any audio it chooses. | • Set every audio stream to “Alternate Audio, Auto-Select, Not Default.” |
| There is no default. The player cannot auto-select any audio. | • Set every audio stream to “Alternate Audio, not Auto-Select.” |
| There is no default. There are specific audios that the player can auto-select. | • Set some audio streams to “Alternate Audio, Auto-Select, Not Default.”  
  • Set some audio streams to “Alternate Audio, not Auto-Select.” |
2. In addition, if you have an audio that is intended as the audio to play when the bandwidth is so low that the video cannot be delivered, then set that audio to “Audio-Only Variant Stream.”

Creating HLS Rendition Groups (Web Interface)

Topics
- Step 1. Create Video-Only Outputs (p. 64)
- Step 3. Create Audio-Only Outputs (p. 64)
- Step 4. Caption-Only Streams (p. 65)
- Step 5. Verify Outputs for the HLS Rendition Group (p. 65)
- Summary of the Steps to Create an HLS Rendition Group (p. 65)
- Example of the Event Output: Creating Caption-Only Streams for an HLS Rendition Group (p. 65)

Step 1. Create Video-Only Outputs

You must create “video-only” outputs. Follow these steps for each video-only output you need:

1. In the AWS Elemental Live web interface, display the Apple HLS output group tab.
2. Click Add Output in order to create an output.
3. Select or create a stream to be associated with that output. For example, Stream 1.
4. In that stream, delete the Default Audio tab. This stream now contains only a video stream.
5. Go back to the Output section associated with this stream and click Advanced. The Audio Rendition Sets field now appears, with a default value of “audio_program.” This field shows in an output only when the associated stream contains only one video stream.
6. Change the Audio Rendition Sets field to specify the rendition group or groups to associate with this video:
   - To associate the video with one rendition group, enter the name of the rendition group.
   - To associate the video with several rendition groups, enter a comma-separated list of the rendition group names. Do not put a space after each comma.

Step 3. Create Audio-Only Outputs

Follow these steps for each audio-only output you need:

1. Click Add Output in order to create an output.
2. Select or create a stream to be associated with that output. For example, Stream 3.
3. In that stream, delete the default Video and Captions tabs. This stream is now an audio stream.
4. Complete the following fields in the Advanced section:
   - **Stream Name**: The wording for the NAME parameter in the manifest, as described in Audio Information for an HLS Output Group with Audio Rendition Group Event (p. 73). This is the audio description that the client player user interface displays. If the description is a language, it should be in that language. For example, “Deutsch “, not “German”.
   - **Language Code**: Optional; complete only if the audio is a language. The wording that is to appear in the LANGUAGE parameter in the manifest should be the language code as per RFC 5646, as described in Audio Information for an HLS Output Group with Audio Rendition Group Event (p. 73). This is the language code that the client player reads.

You can also leave this field blank and check Follow Input Language Code; the language of the audio (specified in Audio Source, a bit higher up on the screen) is detected.
5. Go back to the Output section associated with the first audio stream and click Advanced.

The Audio Group ID field and Audio Track Type now show. These fields show in an output only when the associated stream contains only one audio stream. The Audio Group ID field shows the default value “audio_program.”

(Note that the Audio Only Image field also appears in an audio-only stream. This field has nothing to do with audio rendition groups; it is used to assign an image in a “regular stream that has no video.”)

- Set the Audio Group ID field to specify the audio rendition group that this audio will belong to. For example, “AAC group” or “Dolby group.”
- Set up the Alternate Audio Track Selection field (Audio Track Type field), as described in Step 2. Determine Defaults and Auto-Selection Behavior (p. 62).

Step 4. Caption-Only Streams

If your output includes captions, you must create “captions-only” outputs. Follow these steps for each captions-only output you need:

1. Click Add Output in order to create an output.
2. Select or create a stream to be associated with that output. For example, Stream 4.
3. In that stream, delete the default Video tab and the default Audio tab. This stream now contains only a captions stream.

For more information on setting up captions, see Working with Captions (p. 75).

Step 5. Verify Outputs for the HLS Rendition Group

- Finally, check all your outputs for the HLS output groups and make sure you do not have an output that contains both audio and video. Including such an output may produce a manifest that the client player cannot interpret.

Summary of the Steps to Create an HLS Rendition Group

After these steps, you have:

- One or more video-only outputs. Each output is associated through its Audio Rendition Sets field to one or more audio rendition groups.
- Two or more audio-only outputs. Each output belongs to an audio rendition group based on the value in the Audio Group ID field.
- Optionally, one or more captions-only outputs.

Example of the Event Output: Creating Caption-Only Streams for an HLS Rendition Group

Here is the Output section.
Here is the Streams section.
Creating HLS Rendition Groups (REST API)

The following information assumes that you have read the section called “Creating HLS Rendition Groups (Web Interface)” (p. 64) and are therefore familiar with the construction and association of an output containing video and rendition groups.

Via the REST API, create or modify the event to include the elements and tags in the XML body as described in the following sections.

Topics

- Creating Streams for HLS Rendition Groups in the REST API (p. 67)
- Creating Output Groups for HLS Rendition Groups in the REST API (p. 68)
- Creating Outputs for HLS Output Groups in the REST API (p. 68)
- Sample XML Body For an HLS Output Group with Audio Rendition Group Event (p. 69)

Creating Streams for HLS Rendition Groups in the REST API

- Create as many stream Assembly elements as you require, one for each unique video stream, one for each unique audio stream, and one for each caption stream.
• Each stream_assembly element must contain only of these:
  • One video_description element (plus an optional preset_id tag and name tag), or
  • One audio_description element (plus an optional preset_id tag and name tag), or
  • One caption_description element (plus an optional preset_id tag and name tag).

Creating Output Groups for HLS Rendition Groups in the REST API

• Create as many HLS output groups as desired by creating one output group that has the value "apple_live_group_settings" in its type tag and that contains one apple_live_group_settings element. Set other tags as desired.

Creating Outputs for HLS Output Groups in the REST API

• Within each HLS output group, create as many output elements as required, one for each video stream (plus captions), one for each audio stream, and one for each captions stream.
• Each video output element must contain:
  • container: m3u8
  • extension: m3u8
  • stream_assembly_name: The name of the one stream_assembly to associate with this output. This value matches the value of the name tag in the corresponding stream_assembly_name element.
  • apple_live_settings element that contains:
    • audio_rendition_sets tag: A comma-separated list of the names of the audio rendition groups to associate with this video output to create a set. This value matches the value of the audio_group_id tag in each of the associated audio outputs. For example, “audio_1” in the audio_rendition_sets of this video output matches the “audio_1” in the audio_group_id tag of the associated audio output.
    • Other tags as you require.
• Each audio output element must contain:
  • container: m3u8
  • extension: m3u8
  • stream_assembly_name: The name of the one stream_assembly to associate with this output. This value matches the value of the name tag in the corresponding stream_assembly_name element.
  • apple_live_settings element that contains:
    • audio_group_id: The name of the audio rendition group this audio output belongs to. Specifying a value here creates the rendition group and puts this audio output into that rendition group.
    • audio_track_type: Either “alternate_audio_auto_select_default” or “alternate_audio_auto_select” or “alternate_audio_not_auto_select” or “audio_only_variant_stream”. See the section called “Step 2. Determine Defaults and Auto-Selection Behavior” (p. 62) for information.
    • Other tags as you require.
• Each captions output element must contain:
  • container: m3u8
  • extension: m3u8
  • stream_assembly_name: The name of the one stream_assembly to associate with this output. This value matches the value of the name tag in the corresponding stream_assembly_name element.
  • apple_live_settings element that contains the usual tags as required.
Sample XML Body For an HLS Output Group with Audio Rendition Group Event

This example shows the XML body for an event that contains an HLS output group that includes audio rendition groups.

Following is the <input> element. There are no special rendition group requirements that affect this element.

```xml
<live_event>
  <name>Multi Audio - one video</name>
  <input>
  ...
  ...
  </input>
  ...
</live_event>
```

Following is the <stream_assembly> element for one video. This stream_assembly has the name tag set to “stream_assembly_0” (assigned by default).

```xml
<stream_assembly>
  <name>stream_assembly_0</name>
  ...
  ...
  <video_description>
  ...
  ...
  </video_description>
  <h264_settings>
  ...
  ...
  </h264_settings>
  <codec>h.264</codec>
  ...
  ...
  </stream_assembly>
```

Following is the <stream_assembly> for the first audio. This stream_assembly has the name tag set to “stream_assembly_1” (assigned by default).

```xml
<stream_assembly>
  <name>stream_assembly_1</name>
  <audio_description>
  <follow_input_language_code>false</follow_input_language_code>
  <language_code>eng</language_code>
  <language_description>English</language_description>
  ...
  ...
  </stream_assembly>
```
<stream_assembly>
  <name>stream_assembly_2</name>
  <audio_description>
    <follow_input_language_code>false</follow_input_language_code>
    <language_code>eng</language_code>
    <stream_name>English</stream_name>
    ...
    ...
  <aac_settings>
    ...
    ...
  </aac_settings>
  <codec>aac</codec>
  <audio_source_name>Audio Selector 1</audio_source_name>
</audio_description>
</stream_assembly>

<stream_assembly>
  <name>stream_assembly_3</name>
  <audio_description>
    ...
    ...
  </audio_description>
</stream_assembly>

<stream_assembly>
  <name>stream_assembly_4</name>
  <audio_description>
    ...
    ...
  </audio_description>
</stream_assembly>

Following is the <stream_assembly> for the first caption. This stream_assembly has the name tag set to "stream_assembly_5" (assigned by default).

<stream_assembly>
  <name>stream_assembly_5</name>
  ...
Following are the <stream_assembly> elements for three more captions: stream_assembly_6, stream_assembly_7, and stream_assembly_8.

```xml
<stream_assembly>
  <name>stream_assembly_6</name>
  <caption_description>
    <destination_type>WebVTT</destination_type>
    <language_code>eng</language_code>
    <language_description>English</language_description>
    <order>1</order>
    <caption_source_name>Caption Selector 1</caption_source_name>
  </caption_description>
</stream_assembly>

<stream_assembly>
  <name>stream_assembly_7</name>
  ..
</stream_assembly>

<stream_assembly>
  <name>stream_assembly_8</name>
  ..
</stream_assembly>
```

Following is the <output_group> of type apple_live_group_settings.

```xml
<output_group>
  <apple_live_group_settings>
    ..
  </apple_live_group_settings>
</output_group>
```

Following is the <output> (nested in the HLS output_group element) that is associated with stream_assembly_0 and is therefore a video output. This video is associated with the rendition groups “Audio_aac_hi” and “Audio_aac_lo.”

```xml
<output>
  <extension>m3u8</extension>
  ..
  <apple_live_settings>
    <audio_rendition_sets>Audio_aac_hi,Audio_aac_lo</audio_rendition_sets>
    ..
  </apple_live_settings>
</output>
```
Creating HLS Rendition Groups

Following is the `<output>` (nested in the HLS output_group element) that is associated with `stream_assembly_1` and is therefore an audio output. This audio is part of the rendition group “Audio_aac_hi.”

```
<output>
  <extension>m3u8</extension>
  ...
  ...
  <apple_live_settings>
    <alternate_audio_track_selection>default_audio</alternate_audio_track_selection>
    <audio_group_id>Audio_aac_hi</audio_group_id>
    ...
    ...
  </apple_live_settings>
  <m3u8_settings>
  ...
  ...
  ...
  </m3u8_settings>
  <stream_assembly_name>stream_assembly_1</stream_assembly_name>
  <container>m3u8</container>
</output>
```

More outputs follow, one for each audio stream assembly. Each is part of a rendition group.

```
<output>
  ...
  ...
  ...
</output>
<output>
  ...
  ...
  ...
</output>
<output>
  ...
  ...
  ...
</output>
<output>
  ...
  ...
  ...
</output>
```

Following is the `<output>` (nested in the HLS output_group element) that is associated with `stream_assembly_5` and is therefore a caption output.

```
<output>
  <extension>m3u8</extension>
  ...
  ...
  ...
  <stream_assembly_name>stream_assembly_5</stream_assembly_name>
  <container>m3u8</container>
</output>
```
More outputs follow, one for each caption stream assembly and one for each caption assembly.

```xml
<output>
  
  
  
</output>
<output>
  
  
  
</output>
<output>
  
  
  
</output>
<output>
  
  
  
</output>
</output_group>
</live_event>
```

**Sample HLS Output Group with Audio Rendition Group Event Manifest**

**Video Information for an HLS Output Group with Audio Rendition Group Event**

- There are two video streams, as indicated by the presence of two EXT-STREAM-INF lines.
  - The first video stream has a low bandwidth. As indicated by the AUDIO parameter, it is associated with "audio1."
  - The second video stream has a higher bandwidth. As indicated by the AUDIO parameter, it is associated with "audio2."

**Audio Information for an HLS Output Group with Audio Rendition Group Event**

- There are four audio streams as indicated by the presence of four EXT-X-MEDIA lines with TYPE=AUDIO.
- There are two audio rendition groups, as indicated by the values for the GROUP-ID in each line. The first two lines belong to audio1, the second two to audio2.
- In each audio stream, the values for the various parameters come from these fields in the web interface:
  - TYPE: Always Audio.
  - GROUP-ID: from Audio Group ID field in Output > Advanced.
  - LANGUAGE: from the Language Code field in Stream > Advanced.
  - NAME: from the Stream Name field in Stream > Advanced.
  - AUTOSELECT: from the Audio Track Type in Output > Advanced.
  - DEFAULT: from the Alternate Audio Track Selection field in Output > Advanced.
Capsions Information for an HLS Output Group with Audio Rendition Group Event

- There are two captions streams, as indicated by the presence of two EXT-X-MEDIA lines with TYPE=SUBTITLES.

```plaintext
#EXTM3U
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="audio1",LANGUAGE="eng",NAME="English",AUTOSELECT=YES,DEFAULT=YES,URI="eng1/prog_index.m3u8"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="audio1",LANGUAGE="fre",NAME="français",AUTOSELECT=YES,DEFAULT=NO,URI="fr1/prog_index.m3u8"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="audio2",LANGUAGE="eng",NAME="English",AUTOSELECT=YES,DEFAULT=YES,URI="eng2/prog_index.m3u8"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="audio2",LANGUAGE="fr",NAME="français",AUTOSELECT=YES,DEFAULT=NO,URI="fr2/prog_index.m3u8"

#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",LANGUAGE="eng",NAME="English",DEFAULT=YES,AUTOSELECT=YES,FORCED=NO,URI="1c1.m3u8"
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",LANGUAGE="fra",NAME="French",DEFAULT=YES,AUTOSELECT=YES,FORCED=NO,URI="1c2.m3u8"

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=195023,CODECS="avc1.42e00a,mp4a.40.2",AUDIO="audio1" lo/prog_index.m3u8,SUBTITLES="subs",URI="1c2.m3u8"
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=591680,CODECS="avc1.42e01e,mp4a.40.2",AUDIO="audio2" hi/prog_index.m3u8,URI="1c2.m3u8"
```
AWS Elemental Live Features

This chapter describes the features you can implement in AWS Elemental Live.

Topics
- Working with Captions (p. 75)
- Dynamic Content Switching (p. 109)
- Implementing Graphic Overlay (p. 137)
- Motion Graphic Overlay with SWF (p. 201)
- Output Locking (p. 205)
- SCTE-35 and SCTE-104 Message Processing in AWS Elemental Live (p. 209)
- SCTE-35 Ad Marker EXT-X-DATERANGE (p. 255)
- Ingesting from SMPTE-2110 (p. 256)
- Reference: Supported Captions (p. 257)
- Reference: Supported Codecs and Containers (p. 277)
- Reference: Supported DRM Solutions (p. 289)

Working with Captions

The information in this section assumes that you are familiar with the general steps for creating an event.

You can set up AWS Elemental Live to extract captions when it ingests the source and to include those captions in the output in either the same or a different format. You can include several captions in the output. For example, you can include captions for several languages. You can take a source captions asset and convert it to one format in one output and to another format in a different output.

You perform the setup for captions in your AWS Elemental Live event.

By default, AWS Elemental Live does not ingest any captions (not even captions that are embedded in the video). You must explicitly identify the captions to ingest and the captions to output.

Topics
- Supported Features (p. 75)
- Typical Scenarios (p. 79)
- Setting Up for Captions (p. 81)
- Examples of Implementing Use Cases (p. 102)
- Passing through VBI Data (p. 109)

Supported Features

This section provides information on the various features of captions that AWS Elemental Live supports.

Topics
- Supported Formats (p. 76)
- Supported Captions Categories (p. 77)
- Support for Languages (p. 77)
- Support for Font Styles in Output Captions (p. 78)
• Captions in Events with Multiple Inputs (p. 79)
• Captions and Input Switching Setups (p. 79)

Supported Formats

AWS Elemental Live supports specific formats for captions being ingested, and specific formats in for captions that can be included in outputs. See the section called “Reference: Supported Captions” (p. 257).

Topics

• Format Support by Input Container (p. 76)
• Format Support by Output Container (p. 77)

Format Support by Input Container

This section reviews the options for captions format support by input container, including source captions inside the input container and source captions as an external file.

When Source Captions Are Inside the Input Container

If the source captions are inside the input container, you can include them in the input only if AWS Elemental Live can extract the captions from the container. AWS Elemental Live can extract captions only from specific input container types. The following table shows which captions can be extracted for each input and container type.

<table>
<thead>
<tr>
<th>Type of Input</th>
<th>Container</th>
<th>Captions Can Be Extracted from Container?</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Adobe Flash container</td>
<td></td>
</tr>
<tr>
<td>File</td>
<td>Audio Video Interleave (AVI)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>container</td>
<td></td>
</tr>
<tr>
<td>File</td>
<td>HLS container</td>
<td>Yes</td>
</tr>
<tr>
<td>File</td>
<td>Matroska container</td>
<td></td>
</tr>
<tr>
<td>File</td>
<td>MP4 container</td>
<td>Yes</td>
</tr>
<tr>
<td>File</td>
<td>MPEG Transport Stream (TS)</td>
<td>Yes</td>
</tr>
<tr>
<td>File</td>
<td>MPEG-1 System Stream</td>
<td></td>
</tr>
<tr>
<td>File</td>
<td>MXF container</td>
<td>Yes</td>
</tr>
<tr>
<td>File</td>
<td>No container</td>
<td>Yes</td>
</tr>
<tr>
<td>File</td>
<td>QuickTime container</td>
<td>Yes</td>
</tr>
<tr>
<td>File</td>
<td>Transport Stream</td>
<td>Yes</td>
</tr>
<tr>
<td>File</td>
<td>WMV/ASF container</td>
<td></td>
</tr>
<tr>
<td>Stream</td>
<td>HLS</td>
<td>Yes</td>
</tr>
<tr>
<td>Stream</td>
<td>RTMP</td>
<td>Yes</td>
</tr>
<tr>
<td>Stream</td>
<td>SDI</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Supported Features

<table>
<thead>
<tr>
<th>Type of Input</th>
<th>Container</th>
<th>Captions Can Be Extracted from Container?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream</td>
<td>UDP/RTP</td>
<td>Yes</td>
</tr>
</tbody>
</table>

When Source Captions Are an External File

If the source captions are an external file (sidecar) in one of the supported formats (for example, an SRT file), you can include them in the transcoding regardless of the constraints of the input container.

Format Support by Output Container

- The type of input container. A given input container can contain captions in some formats and not in others.
- The format of the input captions. A given format of captions can be converted to some formats and not to others.
- The type of output containers. A given output container supports some captions formats and not others.

For example, if your input container is an MP4 container and your output is HLS and you want to include Web-VTT captions in the HLS output, you can do so if the MP4 container holds 608 embedded captions. But you cannot include Web-VTT captions if the MP4 container holds ancillary captions.

For complete information about all the supported combinations of input container, input format, and output container, see Reference: Supported Captions (p. 257).

Supported Captions Categories

AWS Elemental Live groups captions formats into several categories. The main categories are the following:

- **Embedded.** The captions are carried inside the video encode. For example, 608 embedded captions.
- **Captions Object.** The captions are in their own "captions encode." They are not part of the video encode. But they are in the same output as their corresponding video and audio encodes. For example, DVB-Sub captions are object-style captions.
- **Sidecar.** The captions are each in their own output, separate from the output that contains the video and audio. The event can contain "captions-only" outputs, for example, one for each language. For example, TTML are sidecar captions.

You need to be aware of captions categories when you set up captions in the output. For more information about all categories and which category each captions format belongs to, see the section called “Step 4: Match Formats to Categories” (p. 88).

Support for Languages

If the source includes captions in multiple languages, you can include multiple languages in the output as follows:

- **Embedded Passthrough.** For any of the embedded source formats, if you specify embedded as the output format, then all languages that are in the input are included in the output. You can't remove any of the languages.
- **Embedded In, Other Out.** For any of the embedded source formats, if you are doing "embedded in, other out," you can specify which languages to extract from the input.
• **Teletext Passthrough.** For teletext sources, if you specify teletext as the output, then all languages (pages) are included in the output. You can't strip out any languages. In fact, the entire teletext content is included in the output; you can't strip out any of the pages. Furthermore, teletext passthrough is supported only in TS outputs.

• **Teletext In, Other Out.** For teletext source, if you are doing “teletext in, other out,” you can specify which languages (teletext pages) to extract and which languages to include in an output.

• **Any Other Combination.** For all other sources, you always specify the language to extract from the input and the language to include in an output, regardless of the source format and output format.

### Support for Font Styles in Output Captions

Depending on the scenario, there are three possibilities for the font style for output captions:

- You can specify the style you want for fonts, including color, outline, and background color.
- The font styles in the input are passed through.
- The font styles are controlled by the downstream player.

### Font Style Options

<table>
<thead>
<tr>
<th>Source Captions</th>
<th>Output Captions</th>
<th>Options for Font Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIB</td>
<td>ARIB</td>
<td>None. The font styles in the input are automatically passed through in the output.</td>
</tr>
<tr>
<td>SCTE-27</td>
<td>SCTE-27</td>
<td>None. The font styles in the input are automatically passed through in the output.</td>
</tr>
<tr>
<td>DVB-Sub</td>
<td>DVB-Sub</td>
<td>None. The font styles in the input are automatically passed through in the output.</td>
</tr>
<tr>
<td>Teletext</td>
<td>Teletext</td>
<td>None. The font styles in the input are automatically passed through in the output.</td>
</tr>
<tr>
<td>Teletext</td>
<td>DVB-Sub</td>
<td>None. The font styles in the input are automatically passed through in the output.</td>
</tr>
<tr>
<td>Any supported captions format</td>
<td>Burn-in</td>
<td>You can specify font styles in the output. If you don’t specify styles, the AWS Elemental Live defaults are used.</td>
</tr>
<tr>
<td>Any supported captions format</td>
<td>DVB-Sub</td>
<td>You can specify font styles in the output. If you don’t specify styles, the AWS Elemental Live defaults are used.</td>
</tr>
<tr>
<td>An Embedded Combination (Embedded, Embedded +SCTE-20, SCTE-20+Embedded)</td>
<td>CCF-TT or TTML</td>
<td>The font information in the source can be copied to the output, or you can let the downstream player determine the font style.</td>
</tr>
</tbody>
</table>
Captions in Events with Multiple Inputs

If your event includes multiple inputs, these rules apply to AWS Elemental Live handling of captions:

- The captions formats in one input can be different from the captions formats in another input. For example, 608 embedded captions might be in one input and teletext might be in another.
- There is no requirement for all the inputs to have captions that are capable of producing the specified captions in any given output.

If the captions from an input cannot produce the specified captions in one of the outputs, the captions will be omitted for the course of that input. The event will not fail. When the event switches to a different input, the captions will be included again if the captions from that input can produce the specified captions in that output.

Captions and Input Switching Setups

Your input might include a backup input that is only switched to if the first input fails. (This feature is called Input Switching or Input Switching with “Hot Hot” backup.) Or your input might include multiple inputs, each with its own backup input – several “input pairs.” The same rules apply as for multiple inputs: the captions in all the inputs must be identical for captions to work smoothly in the output.

Typical Scenarios

Topics
- Use Case: One Input Format to One Output and Not Converted (p. 79)
- Use Case: One Input Format Converted to One Different Format in One Output (p. 80)
- Use Case: One Input Format Converted to Different Formats, One Format for Each Output (p. 80)
- Use Case: One Captions Output Shared by Multiple Video Encodes (p. 81)

The following sections describe some typical scenarios for setting up captions in the event.

These four scenarios demonstrate a range of use cases.

Use Case: One Input Format to One Output and Not Converted

In this case, the input is set up with one format of captions and two or more languages (in the graphic below, you see both English and French.). Assume that you want to maintain the format in the output, that you want to produce only one type of output, and that you want to include all languages in that output.
For example, the input has embedded captions in English and French. You want to produce HLS output that includes embedded captions in both English and French.

Use Case: One Input Format Converted to One Different Format in One Output

The input is set up with one format of captions and two or more languages. You want to convert the captions to a different format in the output. You want to produce only one type of output and include all the languages in that output.

For example, the input has embedded captions in German and French. You want to convert the captions to DVB-Sub and include these captions in both languages in a UDP output.

Use Case: One Input Format Converted to Different Formats, One Format for Each Output

The input is set up with one captions format and two or more languages. Assume that you want to produce several different types of output, and that in each output you want to convert the captions to a different format, and include all the languages.

For example, the input has teletext captions in Czech and Polish. You want to produce an MS Smooth output and an HLS output. In the MS Smooth output, you want to convert both captions to TTML. In the HLS output, you want to convert both captions to WebVTT.
Use Case: One Captions Output Shared by Multiple Video Encodes

This use case deals with captions in an ABR workflow. In this example, one captions output is shared by several video encodes.

For example, assume that there are three video/audio media combinations: one for low-resolution video, one for medium, and one for high. Assume that there is one output captions asset (English and Spanish embedded) that you want to associate with all three video/audio media combinations.

Setting Up for Captions

When you create an event, you must specify the format of the input captions. On the output side, you must specify the desired formats of the captions for each output. When you save the event, AWS Elemental Live validates your choices in terms of whether the specified input format can produce the specified output format, and whether that output format is supported in the specified output type.

Topics

- Step 1: Identify the Source Captions That You Want (p. 81)
- Step 2: Create Captions Selectors (p. 82)
- Step 3: Plan Captions for the Outputs (p. 87)
- Step 4: Match Formats to Categories (p. 88)
- Step 5: Create Captions Encodes (p. 89)

Step 1: Identify the Source Captions That You Want

You must identify the captions that you want to use and assign each to a captions selector. If you don't create any captions selectors, you will not be able to include captions in the output. All the captions will be removed from the media.

To identify the captions you want

1. Identify which captions are in the input (the provider of the input should provide you with this information) and identify which captions are available to you as external files. Identify the captions formats and, for each format, the languages.
2. Identify which of those formats and languages that you want to use.
3. Determine how many captions selectors to create in the input in the event, using the following guidance:
   - For embedded passthrough, create a single captions selector for all languages. All languages are passed through; there is no other option. For details, see the section called “Information for Embedded” (p. 83).
   - For embedded-to-other-format, create one captions selector for each language.
   - For teletext passthrough, create a single captions selector for all languages (in fact, one captions selector for the entire content). All languages (teletext pages) are passed through; there is no other option. For details, see the section called “Information for Teletext” (p. 86).
   - For teletext-to-other-format, create one captions selector for each language.
   - In all other cases, create one captions selector for each language and format combination.
4. You end up with a list of captions selectors to create. For example:
   - Captions Selector 1: teletext captions in Czech
   - Captions Selector 2: teletext captions in Polish

You are not required to use all the languages that are available. You can ignore those you are not interested in.

**Step 2: Create Captions Selectors**

After you have created a list of captions selectors, you can create the captions selectors in the event.

**To create the captions selectors**

1. In the event, in the Input section, choose Advanced.
2. Choose Add Caption Selector.
3. For Source, choose the format of the source captions.
   - To identify SMPTE-TT as the source captions, choose TTML. When AWS Elemental Live ingests the captions, it automatically detects that they are SMPTE-TT.
4. For most formats, more fields appear. For details about a field, choose the Info link next to the field. In addition, see extra information on DVB-Sub or SCTE-27 (p. 82), on Embedded (p. 83), on SCC (p. 85), on SMI, SRT, STL, TTML (p. 85), on teletext (p. 86), or on Null (p. 87).
5. Create more captions selector, as required.

**Information for DVB-Sub or SCTE-27**

This section provides information specific to DVB-Sub or SCTE-27 input captions. It describes the fields that appear when you choose DVB-Sub or SCTE-27 in the Source field in the Caption Selector section of the event. For more context, see the steps earlier in this section.

DVB-Sub and SCTE-27 formats are supported only in TS inputs. You must specify the location of the captions by completing the PID and/or Language code fields in one of these ways.

<table>
<thead>
<tr>
<th>PID</th>
<th>Language Code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified</td>
<td>Blank</td>
<td>Extracts captions from the specified PID.</td>
</tr>
</tbody>
</table>
### Setting Up for Captions

<table>
<thead>
<tr>
<th>PID</th>
<th>Language Code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blank</td>
<td>Specified</td>
<td>Extracts the specified language from whichever PID that languages is in.</td>
</tr>
<tr>
<td>Specified</td>
<td>Specified</td>
<td>Extracts captions from that PID; the language is informational.</td>
</tr>
<tr>
<td>Blank</td>
<td>Blank</td>
<td>Valid only if the source is DVB-Sub and the output is DVB-Sub. With this combination of PID and Language, all input DVB-Sub PIDs will be included in the output. Not valid for SCTE-27.</td>
</tr>
</tbody>
</table>

### Information for Embedded

This section provides information specific to embedded input captions. It describes the fields that appear when you choose Embedded in the Source field in the Caption Selector section of the event. For more context, see the steps earlier in this section.

Read this section if the input captions you have are any of the following: embedded (EIA-608 or CEA-708), embedded+SCTE-20, SCTE-20+embedded, or SCTE-20.

**Note**

For captions in VBI data: If you are extracting embedded captions from the input and using embedded captions in the output, and if the input includes VBI data and you want to include all that data in the output, then do not follow this procedure. Instead, see Passing through VBI Data (p. 109).

### Determining the Number of Captions Selectors Needed

To determine the number of captions selectors you need to created in the event, follow these rules:

- **Embedded Passthrough** – Create only one captions selector. With this scenario, all languages are automatically extracted and are automatically included in the output.
- **Embedded In, Other Out** – If you are setting up embedded-to-other, create one captions selector for each language that you want to include in the output, to a maximum of four selectors.
- **A combination of Embedded passthrough and Embedded conversion** – If you are setting up embedded passthrough in some outputs and embedded-to-other in other outputs, create one captions selector for each language that you want to include in the output, to a maximum of four selectors. Do not worry about a selector for the embedded passthrough output. AWS Elemental Live will extract all the languages for that output, even though no selector exists to explicitly specify this action.

### Completing the Fields in the Captions Selector Group

- **Source:**
  - Choose embedded if the source captions are embedded (EIA-608 or CEA-708), embedded+SCTE-20, or SCTE-20+embedded.
  - Choose SCTE-20 if the source captions are SCTE-20 alone.
Completing the Fields in the CC Channel Number Selector Group

- **CC Channel number**: This field specifies the language to extract. Complete as follows:
  - If you are setting up embedded passthrough only (you are creating only one captions selector for the input embedded captions), this field is ignored, so keep the default.
  - If you are setting up embedded-to-another-format, (you are creating several captions selectors, one for each language), enter the number of the CC instance (from the input) that holds the desired language. For example, if this captions selector is intended to hold the French captions and the French captions are in event 2, enter 2 in this field.

- **Force 608 to 708 Upconvert**: The embedded source captions can be EIA-608 captions, CEA-708 captions, or both EIA-608 and CEA-708. You can specify how you want these captions to be handled when AWS Elemental Live is ingesting content. The following table describes the behavior for various scenarios.

<table>
<thead>
<tr>
<th>EIA-608 in Source</th>
<th>CEA-708 in Source</th>
<th>Convert Field</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>Checked</td>
<td>CEA-708 data is created based on the EIA-608 data. EIA-608 data is added as 608-compatibility bits in the CEA-708 data.</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>Unchecked</td>
<td>Original EIA-608 is preserved.</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Checked</td>
<td>Original CEA-708 is preserved.</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Unchecked</td>
<td>Original CEA-708 is preserved.</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Checked</td>
<td>CEA-708 data is discarded. New CEA-708 data is created based on the EIA-608 data, and EIA-608 data is added as 608-compatibility bits in the CEA-708 data. The new CEA-708 data will not include any CEA-708 formatting features. Not recommended.</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Unchecked</td>
<td>Original EIA-608 is preserved and original CEA-708 is preserved.</td>
</tr>
</tbody>
</table>

- **Use SCTE-20 if Embedded Unavailable**: This field appears only if you set the **Source** to Embedded. If the source captions combine embedded (EIA-608 or CEA-708) and SCTE-20, you might want to set this field to **Auto**. AWS Elemental Live will give preference to the 608/708 embedded captions but will
switch to use the SCTE-20 captions when necessary. If you set this field to Off, AWS Elemental Live will never use the SCTE-20 captions.

**Information for SCC**

This section provides information specific to SCC input captions. It describes the fields that appear when you choose SCC in the **Source** field in the **Caption Selector** section of the event. For more context, see this procedure.

SCC source captions are supplied in a captions file that is external to the video input. You must specify this file.

- **External Caption File**: Specify the location of the file.
- **Time Delta**: Complete this field to adjust the timestamp in the caption file. With the SCC files, the situation sometimes arises where the timestamp in the file for the first captions does not work with the video. The start time for the video/audio always 00:00:00. The start time of the captions may not be 00:00:00 – it may be some completely different, arbitrary time, such as 20:00:15. Assume that, in the video, the first words are spoken at 00:06:15. But given that the start time for the captions file is 20:00:15, then the time for the first caption will be marked as 20:06:30. This time will usually never work with the video. The solution is to adjust the times in the captions file. In this example, subtract 20 hours and 15 seconds (72015 seconds) from the captions file.

Enter a value in this field to push the captions earlier or later:

- Enter a positive number to add to the times in the caption file. For example, enter **15** to add 15 seconds to all the times in the caption file.
- Enter a negative number to subtract from the times in the caption file. For example, enter **-5** to remove 5 seconds from all the times in the caption file.

The format of the times in the captions does not have to match the value in the **Timecode Config** field (in the Input) of the video. The number you enter in this field will simply delay the captions or make the captions play earlier, regardless of the formats.

When using SCC, the video must absolutely have a value in the **Timecode Config** field. Otherwise the captions will not be inserted.

- **Force 608 to 708 Upconvert**: SCC source captions are EIA-608 format and are contained in an external file. The options for converting the caption are the following:
  - Check: To convert the captions to CEA-708 format.
  - Unchecked: To leave the captions unconverted.

**Information for SMI, SMPTE-TT, SRT, STL, TTML**

This section provides information specific to SMI, SMPTE-TT, SRT, STL, and TTML input captions. describes the fields that appear when you choose SCC in the **Source** field in the **Caption Selector** section of the **Create New Live Event** screen. For more context, see the section called “Step 1: Identify Source Captions” (p. 81).

With these formats, the source captions are supplied in a captions file that is external to the video input. You must specify this file.

- **External Caption File**: Specify the location of this file.
- **Time Delta**: Complete this field to adjust the timestamp in the caption file. With the SCC files, the situation sometimes arises where the timestamp in the file for the first captions does not work with the video. With these types of captions, the start time for both the video/audio always 00:00:00.
Assume that, in the video, the first words are spoken at 00:06:15. But in the captions file, this time is marked as 00:06:18, and every other caption is also off by 3 seconds. The solution is to adjust the times in the captions file. In this example, subtract 3 seconds from the captions file.

Enter a value in this field to push the captions earlier or later.
- Enter a positive number to add to the times in the caption file. For example, enter 2 to add 2 seconds to all the times in the caption file.
- Enter a negative number to subtract from the times in the caption file. For example, enter -3 to remove 3 seconds from all the times in the caption file.

Information for Teletext

This section provides information specific to Teletext input captions. It describes the fields that appear when you choose SCC in the Source field in the Caption Selector section of the event. For more context, see the section called “Step 1: Identify Source Captions” (p. 81).

Teletext is a form of data that can contain several types of information, not just captions. Teletext can be present in SDI input, in MXF input, and in TS input, in which case it might be referred to as “DVB teletext.”

You can set up to handle teletext in one of the following ways:
- If you want to extract the entire teletext input, you must set up teletext passthrough. The entire teletext can never be converted to another format. Teletext passthrough is supported only in a TS output.
- You can extract individual captions pages (the captions in a specific language) and convert them to another captions format.
- You cannot extract individual captions pages (the captions in a specific language) and keep them in teletext.

Determining the Number of Captions Selectors Needed

- If you are setting up teletext passthrough captions, create only one captions selector, even if you want to include multiple languages in the output. With this scenario, all languages are automatically extracted and are automatically included in the output.
- If you are setting up teletext-to-other, create one captions selector for each language that you want to include in the output. For example, one selector to extract English teletext, and one selector to extract Swedish teletext.
- If you are setting up teletext passthrough in some outputs and teletext-to-other in other outputs, create individual selectors for the teletext-to-other, one for each language being converted. Do not worry about a selector for the teletext passthrough output. AWS Elemental Live will extract all the data in the teletext, even though there is not a selector to explicitly specify this action.

Completing the Fields in the Captions Selector Group

- **Source**: Choose Teletext.
- **Page**: This field specifies the page of the desired language. Complete as follows:
  - If you are setting up teletext passthrough captions (you are creating only one captions selector for the input captions), leave blank: the value is ignored.
  - If you are converting teletext to another format (you are creating several captions selectors, one for each language), specify the page for the desired language. If you leave this field blank, you get a validation error when you save the event.
Information for Null

The list of sources in the Sources field for the caption selector includes the option Null. This source is not intended for stripping out captions. Instead, it is used for 608 XDS data (p. 101).

Step 3: Plan Captions for the Outputs

If you followed the instructions in the section called “Step 1: Identify Source Captions” (p. 81), you should have a list of the captions formats and languages that will be available for inclusion in the outputs.

You must now plan the captions information for the outputs.

To plan captions for the outputs

1. Identify the types of output media that you plan to create in the event. For example, MS Smooth and HLS.
2. Identify the streams (the combinations of video and audio) that you plan to create for each output media.
3. Map each output to the stream it uses. For example:
   - HLS (Output 1) uses video/audio Stream 1.
   - DASH (Output 2) also uses video/audio Stream 1. (Or it might need its own stream if the video requirements are different.)
4. For each output media, identify which input captions will be converted to which output formats. For example, you might convert teletext captions to TTML for the MS Smooth output media, and those same teletext captions to WebVTT for the HLS output media.

   The output formats that are possible depend on the input formats and the type of output media. See Reference: Supported Captions (p. 257) to determine which output captions are possible given the input format.
5. Identify the languages for each output format:
   - In general, count each language separately.
   - Exception: For embedded passthrough, count all languages as one.
   - Exception: For teletext passthrough, count all languages as one.

The Result

You end up with a list of outputs, and the captions formats and languages for each output. For example:

- MS Smooth output with TTML captions in Czech
- MS Smooth output with TTML captions in Polish
- HLS output with WebVTT captions in Czech
- HLS output with WebVTT captions in Polish.

Planning for Output in Multiple Formats

You can include captions from two or more different formats in an output. For example, you can include both embedded captions and WebVTT captions in an HLS output, to give the downstream system more choices about which captions to use. The only rules for multiple formats are the following:

- The output container must support all the formats. See Reference: Supported Captions (p. 257).
• The font styles in all the captions that are associated with an output must match. This means that the end result must be identical, not that you must use the same option to get that result. For example, all captions that are associated with the output must be white for the first language and blue for the second language.

Managing this style matching can be a little tricky. For information about the font style options, see Support for Font Styles in Output Captions (p. 78).

**Step 4: Match Formats to Categories**

There are different procedures to follow to create captions encodes in the output. The correct procedure depends on the “category” that the output captions belong to. There are five categories of captions, described in the following table.

On the list of outputs that you have created, make a note of the category that each captions option belongs to.

<table>
<thead>
<tr>
<th>Format of Output Captions</th>
<th>Category of This Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ancillary+Embedded</td>
<td>The captions in ancillary format are in the ancillary data in the stream. The embedded captions are embedded in the video.</td>
</tr>
<tr>
<td></td>
<td>To choose Ancillary+Embedded, choose <strong>Embedded</strong> as the <strong>Destination Type</strong> (in the procedure (p. 90)). AWS Elemental Live will automatically produce both Ancillary and embedded.</td>
</tr>
<tr>
<td>ARIB</td>
<td>Object</td>
</tr>
<tr>
<td>Burn-in</td>
<td>Burn-in</td>
</tr>
<tr>
<td>CFF-TT</td>
<td>Object</td>
</tr>
<tr>
<td>DVB-Sub</td>
<td>Object</td>
</tr>
<tr>
<td>Embedded</td>
<td>Embedded</td>
</tr>
<tr>
<td>Embedded+SCTE-20</td>
<td>Embedded</td>
</tr>
<tr>
<td>RTMP CaptionInfo</td>
<td>Object</td>
</tr>
<tr>
<td>RTMP CuePoint</td>
<td>Object</td>
</tr>
<tr>
<td>SCC</td>
<td>Sidecar</td>
</tr>
<tr>
<td>SCTE-20+Embedded</td>
<td>Embedded</td>
</tr>
<tr>
<td>SCTE-27</td>
<td>Object</td>
</tr>
<tr>
<td>SMI</td>
<td>Sidecar</td>
</tr>
<tr>
<td>SMPTE-TT</td>
<td>Sidecar when in Archive</td>
</tr>
<tr>
<td>SMPTE-TT</td>
<td>Stream when in MS Smooth</td>
</tr>
<tr>
<td>SRT</td>
<td>Sidecar</td>
</tr>
<tr>
<td>teletext</td>
<td>Object</td>
</tr>
</tbody>
</table>
### Format of Output Captions

<table>
<thead>
<tr>
<th>Format of Output Captions</th>
<th>Category of This Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTML wrapped in ID3 data</td>
<td>Wrapped in ID3 data</td>
</tr>
<tr>
<td>TTML</td>
<td>Sidecar</td>
</tr>
<tr>
<td>WebVTT</td>
<td>Sidecar</td>
</tr>
</tbody>
</table>

For example, your list of outputs might now look like this:

- MS Smooth output with TTML captions (sidecar) in Czech.
- MS Smooth output with TTML captions (sidecar) in Polish.
- HLS output with WebVTT captions (sidecar) in Czech.
- HLS output with WebVTT captions (sidecar) in Polish.

### Captions Embedded in Video

The captions are carried inside the video encode, which is itself in an output in the output group. Only one captions asset ever exists within that video encode. That single asset might contain captions for several languages.

### Captions Object

The captions are in their own "captions encode" in an output in the output group. They are not part of the video encode. However, they are in the same output as their corresponding video and audio encodes. There might be several captions encodes in the output, for example, for different languages.

### Sidecar

The captions are each in their own output in the output group, separate from the output that contains the video and audio. Each captions output contains only one captions asset (file), meaning that it is a "captions-only" output. The output group might contain several "captions-only" outputs, for example, one for each language in the output group.

### TTML Captions Wrapped in ID3 Data

The captions are converted to TTML and included in ID3 data. (The other way to produce TTML output is as a sidecar.)

### SMPTE-TT in MS Smooth

The captions are handled as a separate stream in MS Smooth.

Note that SMPTE-TT captions for other package types are handled as sidecars. However, for both sidecar handling and stream handling, the procedure for setting up (p. 98) SMPTE-TT captions in the output is identical. AWS Elemental Live will package the SMPTE-TT captions correctly for the package type.

### Burn-in

Here, the captions are converted into text and then overlaid on the picture directly in the video encode. Strictly speaking, once the overlay occurs, these are not really captions because they are indistinguishable from the video.

### Step 5: Create Captions Encodes

Go through the list of outputs you created and set up the captions in each output group, one by one.
Follow the procedure that applies to the format category of the captions output.

Topics
- All Captions except Sidecar or SMPTE-TT in MS Smooth (p. 90)
- Sidecar Captions or SMPTE-TT Captions in MS Smooth (p. 98)
- TTML Captions Wrapped in ID3 Data (p. 100)
- Setting up for 608 XDS Data (p. 101)

All Captions except Sidecar or SMPTE-TT in MS Smooth

Follow this procedure if the format of the captions asset that you want to add belongs to the category of embedded, burn-in, or object. You will set up the captions and video and audio in the same output.

To create captions (not sidecar or SMPTE-TT)

1. On the web interface, on the Event screen, click the appropriate output group.
2. If you have already set up this output group with video and audio, find the outputs where you want to add the captions. Or if you have not set up with video and audio, create a new output in this output group; you can set up the captions now and you can set up the video and audio later.
3. Go to the output, then go to the stream that is associated with that output. For example, go to Stream 1.
4. Click the + beside Caption to add a Caption section.
5. Complete the fields that appear for the selected format. For details about a field, choose the Info link beside the field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Applicability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caption Source</td>
<td>All formats</td>
<td>Select the Caption Selector that you created when you specified the input captions.</td>
</tr>
<tr>
<td>Destination Type</td>
<td>All formats</td>
<td>Select the caption type.</td>
</tr>
</tbody>
</table>
| Pass Style Information | If Destination Type is CCF-TT or TTML. And if the source caption type is an Embedded combination (Embedded, Embedded+SCTE-20, SCTE-20+Embedded), or Teletext, or TTML, or SMPTE-TT, or CCF-TT. | The choices are:  
  - Check this box if you want the style (font, position and so on) of the input captions to be copied.  
  - Leave unchecked if you want a simplified caption style. Some client players work best with a simplified caption style.  
  (For other combinations of source caption types and output caption type, the output is always simplified.) |
| Font style fields      | Destination type is Burn-in or DVB-Sub | For tips about font styles in DVB-Sub or burn-in, see Font Styles for Burn-in or DVB-Sub Output (p. 91). |
### Setting Up for Captions

<table>
<thead>
<tr>
<th>Field</th>
<th>Applicability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>All captions except not for embedded-to-embedded or teletext-to-teletext.</td>
<td>Complete if desired. This information may be useful to or required by a downstream system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For embedded-to-embedded or teletext-to-teletext, leave as Undefined.</td>
</tr>
<tr>
<td>Description</td>
<td>All captions except not for embedded-to-embedded.</td>
<td>This field is auto-completed after you specify the language.</td>
</tr>
<tr>
<td>Use ID3 as Caption Content</td>
<td>Destination type is TTML And if this stream is associated with an MS Smooth output.</td>
<td>Leave unchecked. This field applies only when wrapping TTML captions in ID3; see the section called “TTML Captions Wrapped in ID3 Data” (p. 100).</td>
</tr>
</tbody>
</table>

6. If the output format is embedded and the output group is HLS, you can include captions language information in the manifest. You perform this setup in the output settings (separate from the captions encode). See the section called “Set up the HLS Manifest (Embedded Captions)” (p. 97).

7. If the output format is ARIB or DVB-Sub or SCTE-27, you must perform some extra setup in the output settings (separate from the captions encode). See PIDS for ARIB output (p. 96) or PIDs for DVB-Sub output (p. 97) or PIDs for teletext output (p. 97).

8. You now have a captions encode that is fully defined.

9. Repeat these steps to create captions, as applicable.

10. Go to the output group and output that this stream belongs to. Set the Stream field in that output to match the stream you created.

11. When you are ready, save the event.

If the “Caption Stream Incompatible” message appears, see the section called “Caption Stream Incompatible” Message” (p. 98).

### Font Styles for Burn-in or DVB-Sub Captions

When you set up the captions encode as described in the section called “All Captions except Sidecar or SMPTE-TT in MS Smooth” (p. 90), you can specify the appearance of the captions if the output captions are Burn-in or DVB-Sub. In the following table, the first column shows the field name, the third column specifies how to complete the field, and the third column specifies whether the description applies to Burn-in or DVB-Sub.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Applicability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font</td>
<td>Click <strong>Browse</strong> to find a font file to use. The file must be on a server mounted to the node and must have the extension TTF or TTE. Do not specify a font file if the caption source is embedded or teletext.</td>
<td>Both</td>
</tr>
</tbody>
</table>
## Setting Up for Captions

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Applicability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font Size</td>
<td>Specify <strong>auto</strong> or enter a number. When set to auto, font size will scale depending on the size of the output. Giving a positive integer will specify the exact font size in points.</td>
<td>Both</td>
</tr>
<tr>
<td>Font Resolution</td>
<td>Font resolution in DPI (dots per inch). Range: 96 to 600. Default is 96 dpi.</td>
<td>Both</td>
</tr>
</tbody>
</table>
| Text Justify | For conversions from STL to Burn-in: This field is ignored; the justification specified in the input STL file is always used. For all other conversions to Burn-in:  
  - Centered: If X Position and Y Position are both empty, positions the captions at the bottom center of the video frame.  
    If X Position and Y Position are specified, the captions are offset and then centered across the video frame.  
  - Left: If X Position and Y Position are both empty, positions captions at the bottom left of the video frame.  
    If X Position and Y Position are specified, the captions are offset and then left-aligned. | Burn-in        |
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Applicability</th>
</tr>
</thead>
</table>
| Text Justify  | • Centered: If X Position and Y Position are both empty, positions the captions at the bottom center of the video frame.  
If X Position and Y Position are specified, the captions are offset and then centered across the video frame.  
• Left: If X Position and Y Position are both empty, positions captions at the bottom left of the video frame.  
If X Position and Y Position are specified, the captions are offset and then left-aligned.                                                                                                                   | DVB-Sub       |
| X Position    | For conversions from STL to Burn-in: This field is ignored; the position specified in the input STL file is always used.  
For all other conversions to Burn-in:  
• Offset for the left edge of the caption relative to the horizontal axis of the video frame, in pixels. 0 is the left edge of the video frame. 10 pixels means offset 10 pixels to the right.  
• Empty means 0 offset.                                                                                                                                                                                                                                                                   | Burn-in       |
| X Position    | Offset for the left edge of the caption relative to the horizontal axis of the video frame, in pixels. 0 is the left edge of the video frame. 10 pixels means offset 10 pixels to the right.  
Empty means 0 offset.                                                                                                                                                                                                                                                                                                                          | DVB-Sub       |
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Applicability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y Position</td>
<td>For conversions from STL to Burn-in: This field is ignored; the position specified in the input STL file is always used.</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>For all other conversions to Burn-in:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Offset of the top edge of the caption relative to the vertical axis of the video frame, in pixels. 0 is the top edge of the video frame. 10 pixels means offset 10 pixels from the top.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Empty means position the captions towards the bottom of the output.</td>
<td></td>
</tr>
<tr>
<td>Y Position</td>
<td>Offset of the top edge of the caption relative to the vertical axis of the video frame, in pixels. 0 is the top edge of the video frame. 10 pixels means offset 10 pixels from the top.</td>
<td>DVB-Sub</td>
</tr>
<tr>
<td></td>
<td>Empty means position the captions towards the bottom of the output.</td>
<td></td>
</tr>
<tr>
<td>Fixed Grid</td>
<td>Applies only for conversions from Teletext.</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>• Checked (default): Font is mono-spaced: each character takes up the space horizontal space.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Unchecked: Font is proportionally spaced.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Note that for conversions from STL to Burn-in, the font is always mono-spaced; this information is never in the STL and the value in the field is ignored.)</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Applicability</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Fixed Grid</td>
<td>Applies only for conversions from Teletext.</td>
<td>DVB-Sub</td>
</tr>
<tr>
<td></td>
<td>• Checked (default): Font is mono-spaced: each character takes up the space horizontal space.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Unchecked: Font is proportionally spaced.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Note that for conversions from STL to DVB-Sub, the font is always mono-spaced; this information is never in the STL and the value in the field is ignored.)</td>
<td></td>
</tr>
<tr>
<td>Font Color</td>
<td>For conversions from STL: The font color is taken from the STL file but the value in Font Color is used to override as follows:</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>• If the STL file contains some white text of some &quot;unspecified color&quot; text, then setting the Font Color field changes the white (or unspecified) text. Text in other colors can't be changed. For example, if the file contains white and blue text, the white text is changed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• If the file contains specified text and yellow text, the unspecified text is changed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For all other conversions to Burn-in: Select the desired color.</td>
<td></td>
</tr>
<tr>
<td>Font Color</td>
<td>Select the desired color.</td>
<td>DVB-Sub</td>
</tr>
<tr>
<td>Font Opacity</td>
<td>The opacity for the font color. Range 0 (transparent) to 255 (opaque).</td>
<td>Both</td>
</tr>
<tr>
<td>Background Color</td>
<td>The color for the background rectangle.</td>
<td>Both</td>
</tr>
<tr>
<td>Background Opacity</td>
<td>The opacity for the background rectangle. Range 0 (transparent) to 255 (opaque).</td>
<td>Both</td>
</tr>
</tbody>
</table>
### Name | Description | Applicability
--- | --- | ---
Outline Size | The size for the font outline, in pixels. Range 0 (no outline) to 10. | Both
Outline Color | The color for the font outline. | Both
Shadow Color | The color for the shadow cast by the captions. | Both
Shadow Opacity | The opacity of the shadow, in pixels. Range 0 (transparent) to 255 (opaque). Empty means 0. | Both
Shadow X Offset | The horizontal offset of the shadow, in pixels. A value of -2 results in a shadow offset 2 pixels to the left. A value of 2 results in a shadow offset 2 pixels to the right. | Both
Shadow Y Offset | The vertical offset of the shadow, in pixels. A value of -2 results in a shadow offset 2 pixels above the text. A value of 2 results in a shadow offset 2 pixels above the text. | Both

### Font Styles When You Use the Same Source in Several Outputs

If you are using the same caption source in several Stream sections (in other words, you are selecting the same Caption Selector in the Caption Source field in several Stream sections), then you must set up the font style information identically in each Stream section. If you do not, you will get an error when you save the event.

For example, stream 1 may use Caption Selector 1 with the Destination Type set to Burn-in. And stream 2 may also use Caption Selector 1 with the Destination Type set to Burn-in. You set the font information once in stream 1 and again in stream 2. You must make sure to set up all the font information identically in both streams.

The same rule applies if the output captions are all DVB-Sub.

### Complete the PIDs for ARIB

This section applies when you set up the captions encode as described in the section called “Step 1: Identify Source Captions” (p. 81), if the output group is UDP/TS and the output captions format is ARIB. It describes how to complete the PIDs for the output that contains these captions.

#### To complete the PIDs (ARIB)

1. In the Output section, open the PID Control section.
2. Complete the ARIB Captions field and the ARIB Captions PID field as follows:
ARIB Captions PID Control | ARIB Captions PID | Result
--- | --- | ---
Unchecked | Ignore. | A PID will automatically be assigned during encoding; this value could be any number.
Checked | Type a decimal or hexadecimal. | This PID will be used for the captions.  
| Leave the default (507) | The PID for captions will be 507.  
| Delete the default | A PID will automatically be assigned during encoding; this value could be any number.

Complete the PIDs for DVB-Sub

This section applies when you set up the captions encode as described in the section called “Step 1: Identify Source Captions” (p. 81), if the output group is UDP/TS and the output captions format is DVB-Sub. It describes how to complete the PIDs for the output that contains these captions.

To complete the PIDs (DVB-Sub)

1. In the Output section, open the PID Control section.
2. In the DVB Subtitle PIDs field, enter the PID for the DVB-Sub caption in the stream for this output. Or leave the default.

Complete the PIDs for Teletext

This section applies when you set up the captions encode as described in the section called “Step 1: Identify Source Captions” (p. 81), if the output group is UDP/TS and the output captions format is teletext. It describes how to complete the PIDs for the output that contains these captions.

To complete the PIDs (teletext)

1. In the Output section, open the PID Control section.
2. In the DVB Teletext PID field, enter the PID for the Teletext caption in the stream for this output. Or leave the default.

Set up the HLS Manifest (Embedded Captions)

This section applies when you set up the captions encode as described in the section called “Step 1: Identify Source Captions” (p. 81), if the output group is HLS and the output captions format is embedded. It describes how to include captions language information in the manifest.

To specify language information in the manifest

1. In the HLS output group, go to the output. Click Advanced.
2. Complete Caption Languages as desired:
   - Omit: To omit any CLOSED-CAPTION lines in the manifest.
   - None: To include one CLOSED-CAPTION=None line in the manifest.
   - Insert: To insert one or more lines in the manifest.
3. If you chose Insert, more fields appear. Complete on more sets of fields.
You should complete as many fields as there are languages in this output.

The order in which you enter the languages must match the order of the captions in the source. For example, if the captions are in the order English, then French, then Spanish, then Portuguese, then set up CC1 as English, CC2 as French, and so on. If you do not order them correctly, the captions will be tagged with the wrong languages.

"Caption Stream Incompatible" Message

When you save the event, this validation message might appear:

Stream Caption Destination Type Is Incompatible With XX Output...

Typically, this error will occur because of the following scenario:

- You have two outputs – perhaps HLS and DASH – that will have the same audio and video descriptions, which means you want them to share the same stream:
- You set up the HLS output group and add an Output and Stream 1. You add embedded captions.
- You then set up the DASH output group and add an Output and associate that output with the existing Stream 1.
- The problem is that DASH cannot contain embedded captions. Therefore, when you save the event, you will get the validation message.

The solution to this problem is:

- When you set up the DASH output, instead of associating it with the existing Stream 1, create a new stream (Stream 2)
- In Stream 2, set up the video and audio to be identical to the video and audio in Stream 1.
- For the DASH output, add the captions in the appropriate way.

The result: Assuming that you have set up the video and audio in both streams to be identical, the encoder will notice that they are identical and will in fact encode the video only once and the audio only once. So there will be no extra video encoding load from creating separate streams.

Sidecar Captions or SMPTE-TT Captions in MS Smooth

Follow this procedure if the format of the captions asset that you want to add is a sidecar, as identified in the section called “Step 4: Match Formats to Categories” (p. 88), or if the format is SMPTE-TT for an MS Smooth output.

When you follow this procedure, you set up each captions asset in its own output within the output group. When the event runs, the captions will be set up as sidecars in the output package, except for SMPT-TT captions in MS Smooth, which will be set up as streams in the output package.

To create captions (sidecar and SMPTE-TT)

1. On the web interface, on the Event screen, click the appropriate output group.
2. In the output group, go to the output where you want to add captions.
3. In the upper right corner of the Stream section, click the arrow beside Add a Stream (do not click Add a Stream) and select Add Caption Only Stream. A new (caption-only) stream is added.
4. Complete the fields as shown in the table after this procedure.
5. Repeat these steps to create more sidecar captions in this or another output group, as applicable.
6. When you are ready, save the event.
If the "Caption Stream Incompatible" message appears, see the section called "Caption Stream Incompatible" Message" (p. 98).

<table>
<thead>
<tr>
<th>Field</th>
<th>Applicability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caption Source</td>
<td>All</td>
<td>Select the Caption Selector you created earlier (p. 82).</td>
</tr>
<tr>
<td>Destination Type</td>
<td>All</td>
<td>Select the caption type. This type must be valid for your output type as per the relevant Supported Captions table.</td>
</tr>
</tbody>
</table>
| Framerate      | If the Destination Type is SCC. | Complete this field to ensure that the captions and the video are synchronized in the output.  
Specify a framerate that matches the framerate of the associated video.  
- If the video framerate is 23.97 or 24, choose the corresponding option.  
- If the video framerate is 29.97, choose 29.97 dropframe only if the video has the Video Insertion and Drop Frame Timecode both. |
| Pass-style     | If Destination Type is TTML. And if:  
- The source caption type is TTML, or SMPTE-TT, or CCF-TT.  
- And the output is an Archive output. | Complete as follows:  
- Check this box if you want the style (font, position and so on) of the input captions to be copied.  
- Leave unchecked if you want a simplified caption style. Some client players work best with a simplified caption style.  
(For other combinations of source caption types and output caption type, the output is always simplified.) |
| Font style fields | If the Destination Type is Burn-in | See the table in the section called "Font Styles" (p. 91). |
| Language       | All           | Complete if desired. This information may be useful to or required by a downstream system.                                                 |
**Field** | **Applicability** | **Description**
---|---|---
Description | All | Complete if desired. This information may be useful to or required by a downstream system.

"Caption Stream Incompatible" Message

When you save the event, this validation message might appear:

Stream Caption Destination Type Is Incompatible With XX Output...

Typically, this error will occur because of the following scenario:

- You have two outputs – perhaps HLS and RTMP – that will have the same audio and video descriptions, which means you want them to share the same stream:
- You set up the HLS output group and add an Output and Stream 1. You add embedded captions.
- You then set up the RTMP output group and add an Output and associate that output with the existing Stream 1.
- The problem is that RTMP cannot contain embedded captions. Therefore, when you save the event, you will get the validation message.

The solution to this problem is:

- When you set up the RTMP output, instead of associating it with the existing Stream 1, create a new stream (Stream 2)
- In Stream 2, set up the video and audio to be identical to the video and audio in Stream 1.
- For the RTMP output, add the captions in the appropriate way.

The result: Assuming that you have set up the video and audio in both streams to be identical, the encoder will notice that they are identical and will in fact encode the video only once and the audio only once. So there will be no extra video encoding load from creating separate streams.

**TTML Captions Wrapped in ID3 Data**

Follow this procedure to produce an output that includes TTML captions wrapped in ID3 data. This format is supported only in an MSS output. Unlike unwrapped TTML captions (which you create as described in the section called “Sidecar Captions or SMPTE-TT Captions in MS Smooth” (p. 98)), these captions are included as an ID3 object in the same stream as the video.

**To produce TTML captions wrapped in ID3 data**

1. On the web interface, on the Event screen, click the appropriate output group.
2. In the output group, go to the output where you want to add captions.
3. Identify the stream that is associated with that output. In this example, there are two outputs; the first is associated with stream 1, the second is associated with stream 2.
4. Go to that Stream section. For example, go to Stream 1.
5. Click the + beside Caption to add a Caption section.
6. Complete the fields as shown in the table that follows this procedure.
7. Repeat these steps to add more captions for this output. For example, to add captions in another language.
8. Go to the MSS output group and output that this stream belongs to. Set the Stream field in that output to match the stream you created. For example:

9. When you are ready, save the event.

If the "Caption Stream Incompatible" message appears, see the section called "Caption Stream Incompatible" Message" (p. 100).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caption Source</td>
<td>Select the Caption Selector you created when specifying the input captions (p. 82).</td>
</tr>
<tr>
<td>Destination Type</td>
<td>Select the caption type. This type must be valid for your output type as per the relevant Supported Captions table.</td>
</tr>
<tr>
<td>Pass Style Information</td>
<td>Applicable only if the source caption type is an Embedded combination (Embedded, Embedded +SCTE-20, SCTE-20+Embedded), or Teletext, or TTML, or SMPTE-TT, or CCF-TT. The choices are:</td>
</tr>
<tr>
<td></td>
<td>• Check this box if you want the style (font, position and so on) of the input captions to be copied.</td>
</tr>
<tr>
<td></td>
<td>• Leave unchecked if you want a simplified caption style. Some client players work best with a simplified caption style.</td>
</tr>
<tr>
<td></td>
<td>(For other source caption types, the output is always simplified.)</td>
</tr>
<tr>
<td>Language</td>
<td>Complete if desired. This information may be useful to or required by a downstream system.</td>
</tr>
<tr>
<td>Description</td>
<td>This field is automatically completed after you specify the language.</td>
</tr>
<tr>
<td>Use ID3 as Caption Content</td>
<td>Check this field, to insert the TTML captions into ID3 data.</td>
</tr>
</tbody>
</table>

**Setting up for 608 XDS Data**

If your source content includes 608 XDS data, you can set up the event to include it or strip it from the output.

The Extended Data Services (XDS or EDS) standard is part of EIA-608 and allows for the delivery of ancillary data.

**Note**
You set up handling of this source data for the entire event, so you set up to either include it in every output and stream, or you set up to exclude it from every output and stream.

**To configure handling of this data**

1. In the Input section of the event, click Advanced.
2. Click the **Add Caption Selector** button.
3. Set the source to **Null**.

   You only need to create one Caption Selector for 608 XDS data, regardless of the number of outputs you are creating.

4. If you also want to extract regular captions, create more Caption Selectors according to the regular procedure.
5. In the **Global Processors** section, turn on **608 Extended Data Services** and complete the fields as desired.

**Note**
No setup is required in the captions section of the output or the streams.

### Examples of Implementing Use Cases

The following examples describe how to implement the use cases from the section called “Typical Scenarios” (p. 79).

**Topics**
- **Use Case 1: One Input Format to One Output** (p. 102)
- **Use Case 2: One Input Format Converted to One Different Output Format** (p. 103)
- **Use Case 3: One Input Format Converted to Different Formats, One Format for Each Output** (p. 104)
- **Use Case 4: One Captions Output Shared by Multiple Video Encodes** (p. 106)

### Use Case 1: One Input Format to One Output

This example shows how to implement the first use case (p. 79) from the typical scenarios. The input is set up with one format of captions and two or more languages. Assume that you want to maintain the format in the output and that you want to produce only one type of output and include all the languages in that output.

For example, the input has embedded captions in English and French. You want to produce an HLS output that includes embedded captions in both English and French, plus one video and one audio.

This example illustrates two important features of an embedded passthrough workflow. First, you do not create separate captions selectors; all of the languages are all automatically included. Second, if you are outputting to HLS, you have an opportunity to specify the languages and the order in which they appear.
Event Setup

To maintain the input format on output

1. On the web interface, on the Event screen, for Input settings, choose Add captions selector to create one captions selector. Set Selector settings to Embedded source.
2. In the Output Groups section, create an HLS output group.
3. Create one output and set up the video and audio.
4. In that same output, create one captions asset with the following:
   - Captions selector name: Captions selector 1.
   - Captions settings: One of the Embedded formats.
   - Language code and Language description: Leave blank; with embedded captions, all the languages are included.
5. In the HLS output group, in Captions, in Captions language setting, choose Insert.
6. For HLS settings, in Captions language mappings, choose Add captions language mappings twice (once for each language).
7. Complete the first group of mapping fields with 1, ENG, and English and the second group with 2, FRE, and French.
8. Finish setting up the event and save it.

Use Case 2: One Input Format Converted to One Different Output Format

This example shows how to implement the second use case (p. 80) from the typical scenarios. The input includes two captions languages, and the single output will convert those captions. For example, the input has embedded captions in German and French. You want to produce a UDP output with both captions converted to DVB-Sub, plus one video and one audio.

Event Setup

To convert the input format to another on output

1. On the web interface, on the Event screen, for Input Settings, choose Add captions selector twice, to create Captions selector 1 (for German) and Captions selector 2 (for French). In both cases, set Selector settings to Embedded source.
2. Create a UDP output group.
3. Create one output and set up the video and audio.
4. In this output, choose Add captions to create a captions encode.
Examples of Implementing Use Cases

- **Captions selector name:** Captions selector 1.
- **Captions settings:** DVB-Sub.
- **Language code** and **Language description:** German.
- Other fields: Keep the defaults or complete as desired.

5. Choose **Add captions** again to create another captions encode. Set up this encode for the French captions. Make sure that you set up the font fields for German and French in exactly the same way.

6. Finish setting up the event and save it.

**Use Case 3: One Input Format Converted to Different Formats, One Format for Each Output**

This example shows how to implement the third use case (p. 80) from the typical scenarios. The input is set up with one format of captions and two or more languages. You want to produce several different types of output. In each output, you want to convert the captions to a different format but include all the languages.

For example, the input has teletext captions in Czech and Polish. Assume that you want to produce an MS Smooth output and an HLS output. Assume that in the MS Smooth output, you want to include one video and one audio and you want to convert the captions to TTML. In the HLS output, you want to include one video and one audio and you want to convert the captions to WebVTT.
Event Setup

To convert the input format to a different format for each output

1. On the web interface, on the Event screen, for Input Settings, choose Add captions selector twice, to create the following captions selectors:
   - Captions Selector 1 for teletext Czech. Specify the page that holds the Czech captions.
   - Captions Selector 2 for teletext Polish. Specify the page that holds the Polish captions.

Although you are including this captions in two different outputs (MS Smooth and HLS), you need to extract them from the input only once, so you need to create only one captions selector for each language.

2. Create a MS Smooth output group and configure it as follows:
   - Create one output and set up the video and audio.
   - Create a second output that contains one captions encode and no video or audio encodes and with the following settings:
     - Captions selector name: Captions Selector 1.
     - Captions settings: TTML.
     - Language code and Language description: Czech.
     - Style control: Set as desired.
   - Create a third output that contains one captions encode and no video or audio encodes, with the following settings:
     - Captions selector name: Captions Selector 2.
     - Captions settings: TTML.
     - Language code and Language description: Polish.
   - Other fields: same as the second output (the Czech captions).

3. Create an HLS output group and configure it as follows:
   - Create one output and set up the video and audio.
   - Create a second output that contains one captions encode and no video or audio encodes and with the following settings:
     - Captions selector name: Captions Selector 1.
     - Captions settings: WebVTT.
     - Language code and Language description: Czech.
     - Other fields: Set as desired.
   - Create a third captions output that contains one captions encode and no video or audio encodes and with the following settings:
     - Captions selector name: Captions Selector 2.
     - Captions settings: WebVTT
     - Language code and Language description: Polish.
   - Other fields: same as the second output (the Czech captions).

4. Finish setting up the event and save it.
Use Case 4: One Captions Output Shared by Multiple Video Encodes

This example shows how to set up captions in an ABR workflow. The first setup shows how to set up an ABR workflow when the captions are in the same output as the video, meaning that the captions are either embedded or captions style.

The second setup shows how to set up an ABR workflow when the captions belong to the sidecar category, in which case each captions encode is in its own output.

Event Setup with Embedded or Object-style Captions

This example shows how to implement the fourth use case (p. 81) from the typical scenarios. For example, you want to produce an HLS output with three video encodes (one for low-resolution video, one for medium, one for high) and one audio. You also want to include embedded captions (in English and Spanish) and associate them with all three video encodes.

To use one caption's output in multiple video encodes

1. On the web interface, on the Event screen, for Input Settings, choose Add captions selector to create one captions selector. Set Selector settings to Embedded source.
2. Create an HLS output group.
3. Create one output and set up the video and audio for low-resolution video.
4. In that same output, create one captions asset with the following:
   - Captions selector name: Captions Selector 1.
• **Captions settings**: One of the Embedded formats.

• **Language code and Language description**: Leave blank; with embedded passthrough captions, all the languages are included.

5. Create a second output and set up the video and audio for medium-resolution video.

6. In that same output, create one captions asset with the following:

   • **Captions selector name**: Captions Selector 1.
   
   • **Captions settings**: One of the Embedded formats.
   
   • **Language code and Language description**: Leave blank; with embedded passthrough captions, all the languages are included.

7. Create a third output and set up the video and audio for high-resolution video.

8. In that same output, create one captions asset with the following:

   • **Captions selector name**: Captions Selector 1.
   
   • **Captions settings**: One of the Embedded formats.
   
   • **Language code and Language description**: Leave blank; with embedded passthrough captions, all the languages are included.

9. Finish setting up the event and save it.

**Event Setup with Sidecar Captions**

This example shows an ABR workflow where the captions are in sidecars. For example, you want to produce an MS Smooth output with three video encodes (one for low-resolution video, one for medium, one for high) and one audio. These encodes are in an MS Smooth output. You want to ingest embedded captions (in English and Spanish) and convert them to TTML captions, one for English and one for Spanish.
To set up sidecar captions

1. On the web interface, on the Event screen, for Input Settings, choose Add captions selector twice, to create the following captions selectors:
   - Captions selector 1: for Embedded English.
   - Captions Selector 2: for Embedded Spanish.
2. Create an MS Smooth output group.
3. Create one output that contains one video encode and set it up for low-resolution video.
4. Create a second output that contains one video encode and set it up for medium-resolution video.
5. Create a third output that contains one video encode and set it up for high-resolution video.
6. Create a fourth output that contains one audio encode and no video encode.
7. Create a fifth output that contains one captions encode and no video or audio encodes and with the following settings for the captions encode:
   - Captions selector name: Captions Selector 1.
   - Captions settings: TTML.
   - Language code and Language description: English.
8. Create a sixth output that contains one captions encode and no video or audio encodes, and with the following settings for the captions encode:
   - Captions selector name: Captions Selector 2.
• **Captions settings**: TTML.
• **Language code** and **Language description**: Spanish.

9. Finish setting up the event and save it.

## Passing through VBI Data

AWS Elemental Live supports passthrough of VBI data. You can pass through this data if the following statements are true:

• The input includes VBI data.
• You want to include all that data in the output. This data might include embedded captions.

### To pass through VBI data

1. Create an output for the asset that is to include VBI data.
2. In the **Outputs** section, choose the **Settings** link for the output that contains the video asset.
3. Go to the **Stream** section. Display the **Video** fields. Click **Advanced**. More fields appear.
4. Check the **VBI Passthrough** field.

**Important**  
Do not create a **Captions** object in this output.

All the VBI data (including embedded captions) from the input will be included in the output.

## Dynamic Content Switching

You can set up AWS Elemental Live to process a dynamic playlist consisting of different inputs. Creation of a dynamic playlist lets you use AWS Elemental Live as a playout server/transcoder.

### Typical Use Cases

#### Use Case 1

An event is designed to process live feed from a specific source. The input does not include ad markers.

You create a dynamic playlist in the event to process the live input, then at a given time process one or more file inputs (the ad content), then switch back to the live input, either at a given time or when the last ad content has been processed. The dynamic playlist consists of the single live input “interrupted” periodically by ad content.

#### Use Case 2

An event is designed to process live feed from a specific source. Periodically, the live feed should be replaced by file content (perhaps a movie). Then the same live feed should be resumed.

#### Use Case 3

An event is designed to process live feed from a specific source. However, you may need to cut away from the live feed to process a file input. This cutaway file may be somewhat planned (for example, you may have a “broadcasting will resume shortly” message already prepared), or the cutaway file may be something that is created on the spot to convey an unanticipated public announcement.
You create an event to process the live input. You then either create a dynamic playlist ahead of time to switch to a special file input if required. Or at the last minute you create the entire dynamic playlist and its input, and then switch to it immediately.

**Use Case 4**

An event is designed to process live feed from a specific source. Another event is designed to process live feed from a different source.

Before the dynamic playlist feature was introduced, you could only switch to the other live feed by stopping one event and starting the other. Now, you can merge these two events into one event, with each live feed in the dynamic playlist. You can set up the dynamic playlist to switch from one feed to another either at a scheduled time, or when the operator interrupts.

**Use Case 5**

An event is designed to process file inputs, one after the other, without the event ending.

Before the dynamic playlist feature was introduced, you could add inputs to the event, but only one at a time. Now, you can add as many inputs as you want in one command.

In a variation of this use case, the event may contain only two files; the event can be set up so that as soon as one input ends, the next one starts over again, but with different content specified inside that input.

**Use Case 6**

An event is designed to process file content (perhaps a movie). Periodically, the file content should be replaced by different file content that contains ad content. Then the original file content should be resumed.

You create a dynamic playlist in the event that interleaves sections of the movie with ad content, so: movie, ad content, movie, ad content, movie, and so on. All the inputs are file inputs.

Each time you add the movie file as an input, you include the input clipping tags to create a clip out of a different segment of the movie: 0 to 20 minutes, 20 to 25 minutes, 25 to 40 minutes, and so on. Each time that the movie resumes, it will resume at the desired point.

**The Procedures**

**Topics**

- General Procedure (p. 110)
- Sample Implementations (p. 113)
- Details on Preparing Inputs (p. 115)
- Details on Activating Inputs (p. 116)
- Monitoring Activity via the Web Interface (p. 118)
- Monitoring Activity via the API (p. 118)

**General Procedure**

**Topics**

- Step A: Design the Dynamic Playlist (p. 111)
- Step B: Create and Start the Event (p. 111)
- Step C: Add More Inputs (p. 111)
Step A: Design the Dynamic Playlist

Identify Inputs

1. Identify the inputs and the order in which they are to play.

   - The inputs can be all live assets, all file assets, or a mix of live and file-based assets.
   - The same input can be repeated as many times as you want in the dynamic playlist.

   Generally, it is a good idea to add the inputs to the event in the order in which they will be played. You should do this even if you plan to specify a start time for some or all inputs (below).

Identify Inputs to Prepare Manually

2. Identify inputs that must first be prepared:

   - Live streaming inputs: You must manually prepare each live input before it is due to be played.
   - File inputs: There is no need to manually prepare file inputs, but you may do so if you prefer. Perhaps a reason to prepare is that you know you have to prepare live inputs and you want to follow the same steps for both live and file inputs.

Identify Inputs to Activate Manually

3. Identify inputs that must start at a specific time, rather than simply starting after the previous input ends.

   If you follow the recommended procedure (Step E: below), you will need a placeholder input. Read the procedure in Step E: so that you can prepare this input now.

Step B: Create and Start the Event

4. Create the event with only the first input. (If the event contains more than one input, the others will be deleted when you create the dynamic playlist.)

   Use the REST API Create Event command or use the web interface. See below for tips on setting up the event for the different use cases.

5. Start the event. Use the REST API or the web interface. Processing starts on that single input.

Step C: Add More Inputs

6. Once the event has been started, you can create the dynamic playlist in order to add more inputs to the event.

   There are two ways to add inputs to the dynamic playlist:

   - Use the Add Dynamic Playlist Inputs command to create an array of inputs that are appended to the inputs currently in the event. Up to 29 inputs can be added in this way. For more information, see Add Dynamic Playlist Inputs (p. 120).
• Use the Replace Dynamic List command to create an array of inputs that replaces the existing dynamic playlist (except for the currently running input). You can add any number of inputs this way. For more information, see Replace Dynamic Playlist (p. 122).

Step D: Prepare Inputs

7. If you have identified inputs that must be manually prepared, then as soon as one of those inputs becomes “next in line”, you prepare it by calling Prepare Dynamic Playlist Input (p. 127).

• Time the preparation correctly: For a live input, you must prepare enough time in advance to allow AWS Elemental Live to inspect the entire next-in-line input and decode at least one frame. But you should not prepare it too far in advance because once preparation starts, this is what happens:

AWS Elemental Live inspects the entire input then decodes the first frame in the pipeline. If this input is still not due to run, it discards that frame and decodes the frame that is now first in the pipeline. It discards that frame and decodes the frame that is now first in the pipeline, on and on until the input is due to run. This decoding obviously uses processing resources.

• This timing rule does not apply to file inputs.

• You can optionally specify a prepare time. The input will transition to the Prepare state but no preparing will actually occur until the specified time.

• Keep in mind that you can only manually prepare one input at a time: if you prepare input A, do not prepare the next input until input A is active. If you do, input A may not start as expected. **do json status and check the prepared one is now clear

For more information on the rules around preparation, see Details on Preparing Inputs (p. 115).

Step E: Activate Inputs

8. For each input that must start at a specific time, set the activate time when the input is next in line.

Activating Using Placeholders

Scheduling inputs is a bit tricky because an AWS Elemental Live event must always be encoding: there is no concept of waiting for a frame to arrive. Therefore, if you do not time input B to start immediately after input A ends, then AWS Elemental Live will move to the next input that does not have an activate time (perhaps the input that appears after input B in the XML). But it is very difficult to get this timing between inputs exact enough.

The workaround is to use a “placeholder” input between inputs A and B, as follows:

• Estimate the expected end time of input A.
• Assign an activate time for input B that is as close to the end time without being before it. (In this way, input A is not interrupted.)
• Between input A and input B insert a “placeholder” file input (this input could be a blackout slate, for example). Do not assign a start to this placeholder but do set the loop_source tag to true for this input in order to play the content repeatedly until it is time to return to the live input.

This will be the result: if input A end time does not exactly match the activate time for input B, then the placeholder will play. When the activate time for input B arrives, the placeholder will be interrupted by input B. Input B will start at the scheduled time.

Step F: Continue Adding to the Dynamic Playlist

9. As inputs are run, add more inputs to the event.

• So long as the event is still running, you can continue to add inputs to it; the event will not end.
Procedures

• Add inputs using either the Add Dynamic Playlist Inputs (p. 120) command or the Replace Dynamic List (p. 122) command.

Step G: Interrupt the Currently Active Input

Occasionally, you may need to interrupt the current active input and start a different input (input B). See Implementing Use Case 3 (p. 114) for an example of an unanticipated interruption.

• If necessary, add input B to the dynamic playlist. There are two ways to add:
  • Rebuild the entire playlist, with input B inserted at the top, using the Replace Dynamic Playlist command. The currently active input will not be removed when you do this. For more information, see Replace Dynamic Playlist (p. 122).
  • Append input B to the end of the playlist, using the Add Dynamic Playlist Inputs command. The drawback to this method is that the input appears at the end of the playlist on the web interface, but then gets activated, which may confuse your operators. For more information, see Add Dynamic Playlist Inputs (p. 120).
• If input B is a live input, prepare it using the Prepare Dynamic Playlist command. For more information, see Prepare Dynamic Playlist Input (p. 127).
• Use the Activate Dynamic List Input command (without an activate time) to immediately activate the input. For more information, see Activate Dynamic Playlist Input (p. 128).

Step H: Clean up the Playlist

10. The dynamic playlist is never automatically cleaned up; even after an input has been processed, it remains on the dynamic playlist.

• If you use the Replace Dynamic Playlist command, the entire playlist is replaced, which effectively cleans up old inputs. For more information, see Replace Dynamic Playlist (p. 122).
• If you use the Add Dynamic Playlist Inputs command, then you may want to occasionally use the Delete Dynamic Playlist Input command. For more information, see Delete Dynamic Playlist Input (p. 126).

Step I: Troubleshooting

If the dynamic playlist is not behaving as expected, see the information on preparation and activation starting here (p. 115). You may have broken a preparation or activation rule.

Sample Implementations

Implementing Use Case 1

Use case 1 is described here (p. 109).

1. Create an event that has the live feed as the input.
2. Once the event starts, create a dynamic playlist that consists of the following:
   • Second input – ad content – from file.
   • Third input – live input (identical to first input).
   • Fourth input - ad content – from file.
   • And so on.
3. Immediately set an activate time for the second input. The first input will be interrupted by the second input at this time.
4. After the second input becomes Active, prepare the third input.
When the second input ends, the third input will immediately become Active.

5. After the third input becomes Active, set an activate time for the fourth input. The third input will be interrupted by the fourth input at this time.

**Implementing Use Case 2**

Use case 2 is described here (p. 109).

1. Create an event that has the live feed as the input.
   - Once the event starts, create a dynamic playlist that consists of the following:
     - Second input – ad content – from file.
     - Fourth input and others – file input, as required.
     - Finally, create an input to return to the live feed (same input source as the first input).

2. Immediately set an activate time for the second input in order to interrupt the live feed at the desired time.

   The third input and others will play one after the other, one starting when the previous has completed.

3. When the last file input becomes Active:
   - Optionally set an activate time to return to the live feed. Or omit an activate time and let the live feed resume when the last file has completed.
   - Prepare the live input you are returning to.

**Implementing Use Case 3**

Use case 3 is described here (p. 109).

1. Create an event that has the live feed as the input.

2. Once the event starts, create a dynamic playlist that consists of the following:
   - Second input – a file that displays the desired content. Include the `loop_source` tag for this input in order to play the content repeatedly until it is time to return to the live input.
   - Third input – live input (identical to the first input).

3. If an unanticipated event occurs, switch to the second input: either use the REST API (Activate Dynamic Playlist Input) or let the operator manually activate this input using the web interface control.

4. When you want to resume live input, prepare the third input and then switch to the third input.

5. If another unanticipated event occurs, you can switch again to the second input.

**Implementing Use Case 4**

Use case 4 is described here (p. 110).

1. Create an event that has the live feed as the input.

2. Once the event starts, create a dynamic playlist that consists of the following:
   - Second input – a live input from a different live source.

3. Follow the desired action:
   - Optionally set an activate time to return to the live feed. Or omit an activate time and let the live feed resume when the last file has completed. Or omit the activate time and manually switch to the second input: either use the REST API (Activate Dynamic Playlist Input) or let the operator manually activate the second input using the web interface control.
• Prepare the live input you are returning to.

Implementing Use Case 5

Use case 5 is described here (p. 110).

1. Create an event that has the first file as the input. In the event, set loop_all_inputs to true.
2. Once the event starts, create a dynamic playlist that consists of the following:
   • Second input – a file input.
3. Once the second input has become Active:
   • Modify the first input to point to a different file source. Change other tags as required (for example, the audio selectors).
   • Optionally set an activate time for the first input.

   When the second input has ended, the first input will become Active again.
4. Once the first input has become Active again:
   • Modify the second input to point to a different file source. Change other tags as required (for example, the audio selectors).
   • Optionally set an activate time for the second input.
5. Repeat as required.

Implementing Use Case 6

Use case 6 is described here (p. 110).

1. Create an event that has the live feed as the input.
2. Once the event starts, create a dynamic playlist that consists of the following:
   • Second input – a file input such as a movie. Include the input_clipper tags to clip content. For example, clip it to run from the 0 mark to the 20 minute mark.
   • Third input – ad content – from file.
   • Fourth input – file input identical to the second input. Include the input_clipper tags to clip content, for example, to clip it to run from the 20 minute mark to the 35 minute mark.
   • Fifth input – ad content – from file.
   • Continue switching between the movie and ads.
   • Finally, create a dynamic playlist to return to the live feed.
3. Let each input complete. The next input in the XML will automatically start.
4. When the last file input becomes Active:
   • Optionally set an activate time to return to the live feed. Or omit an activate time and let the live feed resume when the last file has completed.
   • Prepare the live input you are returning to.

Details on Preparing Inputs

Read this section if your inputs are not being prepared as expected.

File Inputs vs Live Streaming Inputs

• Preparing a live input involves a one-time inspection of the stream to determine what audio, video, and data tracks it holds and then continually decoding the frames. So: frame X gets decoded; the input does not become Active so frame X is discarded; frame Y gets decoded; the input does not become
Active so frame Y is discarded; and so on decoding and discarding until finally the input does become Active.

As you can see, this preparation may be expensive in terms of processing power.

- File inputs are automatically prepared just before they are Due-to-be-processed. Preparation of a file input involves decoding the first frame; after that, processing pauses until the input becomes Due-to-be-processed. Preparation of file inputs is not expensive.

**Rules for Preparing**

- The input must not be currently Active.
- Only one input can be in the Prepared state at a time. If you manually prepare input A and then manually prepare input B before A becomes Active, input A will become unprepared.
- An input is put in the Prepared state only when the Prepare command is called on that input. So if a file input that is unprepared is Due-to-be-processed, it transitions directly to the Active state, where it gets both prepared and Active. This means that you can safely prepare a live input several inputs ahead of time, if you want.
- Timing of the manual prepare is important:
  - You should probably prepare a live input only when it is the Next-in-line.
  - You must prepare enough time in advance to allow AWS Elemental Live to inspect the entire third input and decode at least one frame.
  - If the current input is quite long, you may want to wait until it is close to completion before preparing the Next-in-line input.
- If you prepare an input and specify a prepare time, note that the input will transition to the Prepare state but no preparing will occur until the specified time.
- If you both manually prepare an input with a prepare time and manually activate it with an activate time, make sure the prepare time is before the activate time; AWS Elemental Live does not check.
- When you call Prepare Immediately on an input that has previously been set up to be prepared at a time that is still in the future, the input will prepare immediately and the future preparation will be cleared. It will not be prepared both immediately and in the future.

For example, you prepare with a prepare time of 2:00 p.m., then you prepare it immediately, then the input becomes Active and completes at 1:45 p.m. The input will not get prepared again at 2:00 p.m.
- If you prepare input X with a prepare time, any previously prepared input will immediately become unprepared, even though the preparation of input X is still in the future. Although input X is not being prepared, it is still considered to be in the Prepared state; only one input can be Prepared at a time, so the previous input becomes unprepared.

**Details on Activating Inputs**

**Dynamic Playlist Order Rules**

When an input starts to be processed (it is “playing”) that input becomes “active”.

A file input typically moves through these stages:

- Idle and not prepared.
- Next-in-line and not prepared.
- Due-to-be-processed and not prepared.
- Active (prepared and processing).
- Idle (completed)
A live input typically moves through these stages:

- Idle and not prepared.
- Next-in-line and prepared.
- Due-to-be-processed and prepared.
- Active (processing).
- Idle (completed)

**Idle**

Any number of inputs can be idle.

The list of idle inputs includes inputs that have completed and those that have not been processed.

**Next-in-Line**

Only one input can be Next-in-line. This is the input that is either:

- The input that is after the currently active input, based on the order in which the inputs appear in the event XML and on the web interface.
- The input that has an activate time that exactly aligns with the end time of the current input.

Ideally, you should order the inputs in the XML so that an input that has an activate time also appears in its natural order. For example, it does not appear at the end of the list; this placement will cause confusion to your operators.

A next-in-line input may be prepared or unprepared.

"Due-to-be-processed" Input

The input that is currently Next-in-line becomes Due-to-be-processed at the moment that the currently Active input completes.

This input then instantaneously transitions to Active.

**Active Input**

This is the input that is being processed. An input becomes active as follows:

- If the input is a live input has been prepared, it will be activated and processing will start seamlessly.
- If the input is a live input has not yet been prepared, it will still be activated; it will not be skipped. Instead, it will first be prepared, then activated. There will be a noticeable delay while the input is prepared.
- If the input is a file input, it will be prepared and activated.

**Rules for Activating**

- When you call Activate Immediately on an input, the currently Activated input will immediately transition to Idle and processing of that input will stop.
- When you call Activate Immediately on an input that has previously been set up to activate at a time that is still in the future, the input will activate immediately and the future activation will be cleared. It will not be activated both immediately and in the future.
- When you call Activate with Specified Time and then call Prepare, the activate time is not cleared: the input will be prepared (either immediately or later, depending on what you set up), then the input will be activated at the specified time.
• If you call Activate with Specified Time on an input that has not been prepared, the input will first be
prepared (at the specified time), then it will be activated. There will be a noticeable delay in activation.
In other words, activating with a specified time does not automatically set up any prepare schedule.
• If you both manually prepare an input with a prepare time and manually activate it with an activate
time, make sure the prepare time is before the activate time; AWS Elemental Live does not check.

Monitoring Activity via the Web Interface

The operator can monitor dynamic playlist activity via the AWS Elemental Live web interface.

1. On the web interface, display the Event Control tab.
2. If the blue button specifies Control Panel, then the Details panel is currently displayed. Click the
Control Panel button.
3. On the Control Panel, click Input Controls (below the Preview panel) to expand that section. The
dynamic playlist appears.

Status Information

<table>
<thead>
<tr>
<th>Input Background</th>
<th>Icon in Control Column</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Spinner icon</td>
<td>Active</td>
</tr>
<tr>
<td>Green</td>
<td>Arrow icon</td>
<td>Prepared</td>
</tr>
<tr>
<td>Brown</td>
<td>Arrow icon</td>
<td>Being prepared</td>
</tr>
<tr>
<td>Gray</td>
<td>Arrow icon</td>
<td>Idle</td>
</tr>
</tbody>
</table>

The orange numbers down the left side are on-screen numbers, for display purposes only.
The numbers in the ID column are the REST IDs of the inputs.

Controls

The operator can click the triangle to switch to that input. The input will become Active. Processing will
stop on the current Active input.

Monitoring Activity via the API

Fine points and pitfalls

You can obtain a list of inputs currently in the dynamic playlist using the Get Event (p. 123) command
or the Get Event Status (p. 130) command.

You can obtain information on the status of each input in the dynamic playlist using the Get Event Status
command. For more information, see Get Event Status (p. 130).
Using the REST API

Topics
- List of Commands (p. 119)
- Add Dynamic Playlist Inputs (p. 120)
- Replace Dynamic Playlist (p. 122)
- Get Event (p. 123)
- Modify One Dynamic Playlist Input (p. 125)
- Delete Dynamic Playlist Input (p. 126)
- Prepare Dynamic Playlist Input (p. 127)
- Activate Dynamic Playlist Input (p. 128)
- Get Event Status (p. 130)

List of Commands

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Action</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add dynamic playlist inputs</td>
<td>POST</td>
<td>/live_events/&lt;event ID&gt;/inputs</td>
<td>In the specified event (which must be currently running), add the specified input or inputs to the end of the dynamic playlist.</td>
</tr>
<tr>
<td>Replace dynamic playlist</td>
<td>POST</td>
<td>/live_events/&lt;event ID&gt;/playlist</td>
<td>In the specified event (which must be currently running), remove all non-Active inputs from the dynamic playlist and append the specified input or inputs. After this command, only the Active input remains from the original dynamic playlist.</td>
</tr>
<tr>
<td>Get event</td>
<td>GET</td>
<td>/live_events/&lt;event ID&gt;</td>
<td>Gets the contents of the event, including the list of inputs.</td>
</tr>
<tr>
<td>Modify dynamic playlist input</td>
<td>PUT</td>
<td>/live_events/&lt;event ID&gt;/inputs/ &lt;input ID&gt;</td>
<td>In the specified event (which must be currently running), modify the specified dynamic playlist input (which must be non-Active).</td>
</tr>
<tr>
<td></td>
<td>PUT</td>
<td>/live_events/&lt;event ID&gt;/inputs/ by_label/</td>
<td>In the specified event (which must be currently running), modify the specified dynamic playlist input (which must be non-Active).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;input_label&gt;</td>
<td></td>
</tr>
</tbody>
</table>
### Add Dynamic Playlist Inputs

In the specified event (which must be currently running), add the specified input or inputs (maximum 29 inputs for a total of 30, including the Active input) to the end of the dynamic playlist.

### HTTP Request and Response

#### HTTP URL

<table>
<thead>
<tr>
<th>Action</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/live_events/&lt;event ID&gt;/inputs</td>
<td>In the specified event (which must be currently running), add the specified input or inputs (maximum 29 inputs for a total of 30, including the Active input) to the end of the dynamic playlist.</td>
</tr>
<tr>
<td>POST</td>
<td>/live_events/&lt;event ID&gt;/activate_input</td>
<td>In the specified event (which must be currently running), activate the specified dynamic playlist input either at the specified time or immediately.</td>
</tr>
<tr>
<td>POST</td>
<td>/live_events/&lt;event ID&gt;/prepare_input</td>
<td>In the specified event (which must be currently running), prepare the specified dynamic playlist input and optionally activate encoding at the specified time or immediately.</td>
</tr>
<tr>
<td>GET</td>
<td>/live_events/&lt;event ID&gt;/status</td>
<td>Gets the status of the specified event, including information on the stage and state of each input.</td>
</tr>
</tbody>
</table>

**XML content consisting of one:**

```xml
<Root/>
```
One inputs element that contains:

One or more input elements that each contains one or more of the regular input tags. See Elements and Tags in an Event Input XML (p. 132).

Tips for Elements and Tags

Here are some notes on tags of particular interest:

- **input_label**: It is recommended that you include the input_label tag. If you include this tag, you can reference the input later on using this label (rather than using the input ID, which you first have to query for using Get Event).

  An input label must be unique among all the inputs in the event when you add inputs – the existing inputs and the new inputs.

  If you add an input with label X and later remove the input, you can add another input with label X, even if the input actually has different content. AWS Elemental Live does not track the content, it just enforces the rule for label uniqueness at a given time.

- **loop_source**: This tag can be used to loop the dynamic playlist. If you set this tag to true for input X, it is a good idea to set an activate time for the next intended input, otherwise input X will process forever. See Activating Using Placeholders (p. 112) for a typical use case for loop_source.

  When this tag is true and AWS Elemental Live is at the last input listed in the event XML, the first input in the dynamic playlist may be considered as the Next-in-line input.

- **order**: This tag is ignored in determining the dynamic playlist order.

- For the input source, include one of: network_input or device_input or router_input or file_input. Different inputs can have a different source.

- **audio_selector, caption_selector**: All inputs must have the same number of these elements: none or more than one. So if the first input has two audio_selector elements, all inputs must have two audio_selectors. This rule applies to the lifetime of the event: as soon as it starts, the count of audio_selector and caption_selectors is fixed and cannot vary.

- **input_clipping**: Can be used to process only a specific section of a live or file input.

Response

200 OK for a successful request.

Example

This request adds two inputs to the event with the ID 31. The first input is a live input, the second is a file input.

```xml
POST http://10.4.136.92/live_events/31/inputs

<inputs>
  <input>
    <input_label>movie08E45_section_1</input_label>
    <loop_source>false</loop_source>
    <network_input>
      <quad>false</quad>
      <uri>udp://10.0.0.1:5005</uri>
    </network_input>
    <audio_selector>
      <default_selection>true</default_selection>
      <track>1</track>
    </audio_selector>
  </input>
</inputs>
```
Replace Dynamic Playlist

In the specified event (which must be currently running), remove all non-Active inputs from the dynamic playlist and append the specified file input or inputs. After this command, only the Active input remains from the original dynamic playlist; the rest of the dynamic playlist consists of new inputs.

The dynamic playlist can only consist of file inputs; if you want to add live inputs or live and file inputs, use Add Dynamic Playlist Inputs.

There is no maximum to the number of inputs that can be added using this command, and there is no maximum to the number inputs that result in the event. For example, you can replace the 15 inputs currently in the event with 900 new inputs. Compare this lack of restrictions to the restrictions that apply with Add Dynamic Playlist Inputs.

You can use Replace Dynamic Playlist to clear the dynamic playlist: create a Body consisting of an empty inputs element. **file only

HTTP Request and Response

HTTP URL

POST http://<Live IP address>/live_events/<event ID>/playlist

Body of HTTP

XML content consisting of:

- One inputs element that contains:
  - One or more input elements that each contains one or more of the regular input tags. See Elements and Tags in an Event Input XML (p. 132).

Response

200 OK for a successful request.

Example

This request specifies two inputs to the event with the ID 31. The first input is a live input, the second is a file input. This list of inputs will replace the current dynamic playlist, not including the currently active input.

POST http://10.4.136.92/live_events/31/inputs **s/b playlist endpoint

-----------------------------
<input>
  <input_label>enigmatic_car_ad</input_label>
  <loop_source></loop_source>
  <file_input>
    <uri>/data/server/ad13978.mp4</uri>
  </file_input>
  <audio_selector>
    <default_selection>true</default_selection>
    <track>1</track>
  </audio_selector>
</input>
</inputs>
Get Event

Get a list of the inputs in the specified event.

There is no explicit command to get the dynamic playlist. But you can get the event in order to get information about the dynamic playlist.

Get Event is useful for obtaining the IDs of the inputs in the dynamic playlist and for parsing for the order in which they are listed in the XML.

Get Event includes the status of each input in these tags:

- active tag. Possible values are true and false.
- status tag. Possible values are preprocessing, pending, running, postprocessing, complete.
- active_input_id tag. Only the currently active input has this tag. For that input, the active tag and the input_ID specify the same value.

Get Event does not provide information about the prepare time or activate time on inputs on which you explicitly called Prepare with Specified Time or Activate with Specified Time. That information cannot be retrieved from AWS Elemental Live; you must maintain the schedule outside of AWS Elemental Live.

HTTP Request and Response

HTTP URL

GET http://<Live IP address>/live_events/<event ID>

Response

XML content consisting of one event element that contains:

- Various general tags.
• One or more input elements that each contain:
  • A unique ID tag.
  • A unique input_label tag (optional).
  • A status tag.
  • An input element: complete network_input or device_input or router_input or file_input.
  • A video_selector element.
  • An optional audio_selector element.
  • An optional caption_selector element.
• Other elements relating to input.
• Other elements relating to outputs.

**Example**

**Request**

This request gets the event with the ID 31.

```
GET http://10.4.136.92/live_events/31
```

**Response**

The event contains three inputs, with IDs 64, 98, 99, and with input_label tags “movie08E45_section_1”, “enigmatic_car_ad” and “best_trowel_ad”.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<live_event; href="/live_events/31" product="Elemental Live" version="2.17.0.12345">
  <id>31</id>
  ...
  <input>
    <active>false</active>
    <id>64</id>
    <input_label>movie08E45_section_1</input_label>
    <loop_source>false</loop_source>
    <status>pending</status>
    ...
    <network_input>
      <id>4</id>
      ...
      <uri>udp://10.0.0.1:5005</uri>
    </network_input>
    <video_selector>
      <id>2</id>
      ...
    </video_selector>
    <audio_selector>
      <id>2</id>
      ...
    </audio_selector>
  </input>
  ...
</live_event>
```
Modify One Dynamic Playlist Input

In the specified event (which must be currently running), modify the specified dynamic playlist input (which must be non-Active).

The input can be modified in any way, so long as it follows the rules for inputs in a dynamic playlist, as described in Tips for Elements and Tags (p. 121).

HTTP Request and Response

HTTP URL

To specify the input by its REST ID:

| PUT http://<Live IP address>/live_events/<event ID>/inputs/<input id> |

To specify the input by the input label:

| POST http://<Live IP address>/live_events/<event ID>/inputs/by_label/<input_label> |

The input label is the value in the <input_label>; you may have included this tag when you first created the input. If you did not specify an input label, you cannot use this signature to modify.

Body of HTTP

XML content consisting of one input element that each contains only the tags to modify.
Response

200 OK for a successful request.

Example

Request

This request modifies the input with the ID 28 that is in the event with the ID 31. It modifies the input so to point to a different asset.

```
PUT http://10.4.136.92/live_events/31/inputs/28
-----------------------------------------------
<input>
  <network_input>
    <uri>udp://10.0.0.1:5005</uri>
  </network_input>
</input>
```

Delete Dynamic Playlist Input

In the specified event (which must be currently running), delete the specified non-Active input from the dynamic playlist.

You can also delete the dynamic playlist using Replace Dynamic Playlist with an empty inputs in the Body. For more information, see Replace Dynamic Playlist (p. 122).

HTTP Request and Response

HTTP URL

To specify the input by its REST ID:

```
DELETE http://<Live IP address>/live_events/<event ID>/inputs/<input ID>
```

To specify the input by the input label:

```
DELETE http://<Live IP address>/live_events/<event ID>/inputs/by_label/<input label>
```

The input label is the value in the `<input_label>`; you may have included this tag when you first created the input. If you did not specify an input label, you cannot use this signature to delete.

Response

200 OK for a successful request.

Example

Request

This request deletes the input with the label “curling_83399” that is in the event with the ID 31.

```
DELETE http://10.4.136.92/live_events/31/inputs/by_label/83399
```

Version 2.17

126
Prepare Dynamic Playlist Input

In the specified event (which must be currently running), manually prepare the specified dynamic playlist input either at the specified time or immediately.

The input being prepared must not be already Active.

HTTP Request and Response

HTTP URL

POST http://<Live IP address>/live_events/<event ID>/prepare_input

Body of HTTP

XML content consisting of one of the following:

• One input_id tag that contains the ID of the input to prepare.

Or

• One input_label tag that contains the input_label of the input to prepare.

Or

• One prepare_input element that contains:
  • One input_id tag that contains the ID of the input to prepare or one input_label tag that contains the input_label of the input to prepare.
  • One utc_time tag that contains the time at which to prepare the input, in UTC time, down to the seconds and (optionally) fractional seconds.

Response

The entire <input> element for the specified input.

Example 1

Request

In the event with the ID 31, prepare the input with the label “live_news_feed”.

POST http://10.4.136.92/live_events/31/prepare_input

<?xml version="1.0" encoding="UTF-8"?>
<input>
  <input_id>live_news_feed</input_id>
</input>

Response

The response returns the entire <input> element. In this example, this input has the ID 194.

<?xml version="1.0" encoding="UTF-8"?>
<input>
  <active>true</active>
  <id>194</id>
  <input_label>live_news_feed</input_label>
</input>
Example 2

In the event with the ID 31, prepare the input with the ID 103 and activate the input at 2015123T235959.999:

```xml
<prepare_input>
  <input_id>103</input_id>
  <utc_time>2015123T235959.999</utc_time>
</prepare_input>
```

Activate Dynamic Playlist Input

In the specified event (which must be currently running), activate the specified dynamic playlist input either at the specified time or immediately.

The input being prepared must not be already Active.

HTTP Request and Response

HTTP URL

POST http://<Live IP address>/live_events/<event ID>/activate_input

Body of HTTP

XML content consisting of one of the following:

- One input_id tag that contains the ID of the input to activate.

Or
• One `input_label` tag that contains the `input_label` of the input to activate.

Or

• One `activate_input` element that contains:
  • One `input_id` tag that contains the ID of the input to prepare or one `input_label` tag that contains the `input_label` of the input to prepare.
  • One `utc_time` tag that contains the time at which to activate the input, in UTC time, down to the seconds and (optionally) fractional seconds.

Response

The entire `<input>` element for the specified input.

Example 1

Request

In the event with the ID 31, activate the input with the `input_label"syndicated_show_231"`.

```
POST http://10.4.136.92/live_events/31/activate_input
----------------------------------------------------
<input_label>syndicated_show_231</input_label>
```

Response

The response returns the entire `<input>` element. In this example, this input has the ID 64.

```
<?xml version="1.0" encoding="UTF-8"?>
<input>
  <active>true</active>
  <id>64</id>
  <input_label>syndicated_show_231</input_label>
  <loop_source>false</loop_source>
  <status>pending</status>
    ...
  <file_input>
    <id>206</id>
    <uri>/data/server/13978.mp4</uri>
  </file_input>
  <video_selector>
    <id>2</id>
      ...
  </video_selector>
  <audio_selector>
    <id>2</id>
      ...
  </audio_selector>
  <input_info>
    ...
  </input_info>
</input>
```
Example 2

In the event with the ID 31, activate the input with the ID 103 and activate the input at 2015123T235959.999:

```xml
<activate_input>
  <input_id>103</input_id>
  <utc_time>2015123T235959.999</utc_time>
</activate_input>
```

Get Event Status

Get the status of the inputs in the specified event.

There is no explicit command to get the status of the inputs in the dynamic playlist. But you can get the event status in order to get status information about the dynamic playlist.

Warning: Get Event Status only works with the JSON format. Therefore, the request header must include Accept: application/json and Content-type:application/json

HTTP Request and Response

HTTP URL

GET http://<Live IP address>/live_events/<event ID>/status

Response

JSON content consisting of one live_event element that contains:

- One outputs element and one output_groups element that each includes various tags.
- One active_input tag that contains the REST ID of the currently Active input.
- One inputs element that lists the inputs. The inputs appear in the order in which they were originally listed when the dynamic playlist was created. The following information appears for each input:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Integer</td>
<td>The unique REST ID for this input</td>
</tr>
<tr>
<td>state</td>
<td>String</td>
<td>The state of the input:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• clear: The input is Activated or Prepared.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• quarantined: The input is either starting or is recovering from a failure condition.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• pending: The input is Idle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• errored: A failure condition (as defined by the failure_condition tag in the event XML) has been triggered.</td>
</tr>
<tr>
<td>input_label</td>
<td>String</td>
<td>The input label, if one was created.</td>
</tr>
</tbody>
</table>
Tag | Value | Description
--- | --- | ---
uri | String | The URI for a file input.

**Stage and State**

The state tag and active_input tag can provide some information about the stage and state of each input:

<table>
<thead>
<tr>
<th>state Tag</th>
<th>active_input Tag</th>
<th>State and State</th>
</tr>
</thead>
</table>
| pending | Does not specify this input | The input is one of:
• Idle and Unprepared
• Next-in-line and Unprepared |
| clear | Does not specify this input | The input is one of:
• Idle and Prepared
• Active
• Next-in-line and Prepared |
| clear | Specifies this input | The input is:
• Active |
| quarantined | Specifies this input | The input is:
• Active |
| errored | Specifies this input | A failure condition (as defined by the failure_condition tag in the event XML) has been triggered. |

Note that you cannot determine whether the input is Next-in-line from the response to this Get. You must maintain that information outside of AWS Elemental Live.

**Example**

```
GET http://10.4.136.92/live_events/47/status
{
    "live_event": {
        "id": "47",
        "status": "running",
        "outputs": [
            .
            .
            .
        ],
        "output_groups": [
            .
            .
            .
        ],
        "active_input": "533761",
```

Version 2.17
131
Elements and Tags

This section lists all the elements and tags that could appear in the input element of an event XML. This section does not include exhaustive information about rules for if and when an element or tag can be included. For tips on some tags, see Add Dynamic Playlist Inputs (p. 120).

```
<product></product>
<input>
  <deblock_enable></deblock_enable>
  <deblock_strength></deblock_strength>
</input>
```

Version 2.17
132
<error_clear_time></error_clear_time>
<failback_rule></failback_rule>
<hot_backup_pair></hot_backup_pair>
<input_label></input_label>
<loop_source></loop_source>
<name></name>
<no_psi></no_psi>
<order></order>
<program_id></program_id>
<service_name></service_name>
<service_provider_name></service_provider_name>
<image_inserter>
  <image_x></image_x>
  <image_y></image_y>
  <opacity></opacity>
  <image_inserter_input>
    <uri></uri>
    <username></username>
    <password></password>
    <interface></interface>
  </image_inserter_input>
</image_inserter>
<network_input>
  <enable_fec_rx></enable_fec_rx>
  <interface ></interface >
  <password></password>
  <quad></quad>
  <udp_igmp_source></udp_igmp_source>
  <uri></uri>
  <username></username>
</network_input>
<device_input>
  <channel></channel>
  <channel_type></channel_type>
  <device_id></device_id>
  <device_name></device_name>
  <device_number></device_number>
  <device_type></device_type>
  <input_format></input_format>
  <fec_settings>
    <udp_igmp_source></udp_igmp_source>
    <uri></uri>
  </fec_settings>
  <hdmi_settings>
    <input_format></input_format>
  </hdmi_settings>
  <sdi_settings>
    <input_format></input_format>
  </sdi_settings>
</device_input>
<router_input>
  <input_number></input_number>
  <input_number_end></input_number_end>
  <quad></quad>
  <router_ip></router_ip>
  <router_type></router_type>
</router_input>
<file_input>
  <certificate_file></certificate_file>
  <interface></interface>
  <password></password>
  <uri></uri>
  <username></username>
</file_input>
<failover_condition>
<description></description>
<duration></duration>
<order></order>
</failover_condition>
<video_selector>
<color_space></color_space>
<default_afd></default_afd>
<name></name>
<order></order>
<pid></pid>
</program_id></program_id>
</video_selector>
<audio_selector>
<default_selection></default_selection>
<external_audio_file_input></external_audio_file_input>
<infer_external_filename></infer_external_filename>
<language_code></language_code>
<name></name>
<offset></offset>
<order></order>
<pid></pid>
</program_selection></program_selection>
<strict_language_selection></strict_language_selection>
<strict_pid_option></strict_pid_option>
<track></track>
<unwrap_smpte337></unwrap_smpte337>
</audio_selector>
<audio_selector_group>
<audio_selector_name></audio_selector_name>
</name></name>
</audio_selector_group>
<caption_selector>
<order></order>
<source_type></source_type>
<language_code></language_code>
<embedded_source_settings>
<autodetect_scte20></autodetect_scte20>
<source_608_channel_number></source_608_channel_number>
<source_608_track_number></source_608_track_number>
<upconvert_608_to_708></upconvert_608_to_708>
</embedded_source_settings>
<file_source_settings>
<time_delta></time_delta>
<upconvert_608_to_708></upconvert_608_to_708>
<source_file>
<certificate_file></certificate_file>
</interface></interface>
</password></password>
</uri></uri>
</source_file>
</file_source_settings>
<teletext_source_settings>
<page_number></page_number>
</teletext_source_settings>
<dvb_sub_source_settings>
<pid></pid>
</dvb_sub_source_settings>
<scte27_source_settings>
<pid></pid>
</scte27_source_settings>
</caption_selector>
<input_clipping>
<end_timecode></end_timecode>
<order></order>
<start_timecode></start_timecode>
### Broken down into individual components, you can see:

<table>
<thead>
<tr>
<th>Function of code excerpt</th>
<th>Example code excerpt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Various tags</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;href&gt;&lt;/href&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;version&gt;&lt;/version&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;product&gt;&lt;/product&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;input&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;deblock_enable&gt;&lt;/deblock_enable&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;deblock_strength&gt;&lt;/deblock_strength&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;error_clear_time&gt;&lt;/error_clear_time&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;fallback_rule&gt;&lt;/fallback_rule&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;hot_backup_pair&gt;&lt;/hot_backup_pair&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;input_label&gt;&lt;/input_label&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;loop_source&gt;&lt;/loop_source&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;name&gt;&lt;/name&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;no_psi&gt;&lt;/no_psi&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;order&gt;&lt;/order&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;program_id&gt;&lt;/program_id&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;service_name&gt;&lt;/service_name&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;service_provider_name&gt;&lt;/service_provider_name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Image inserter for this input</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;image_inserter&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;image_x&gt;&lt;/image_x&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;image_y&gt;&lt;/image_y&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;opacity&gt;&lt;/opacity&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;image_inserter_input&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;uri&gt;&lt;/uri&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;username&gt;&lt;/username&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;password&gt;&lt;/password&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;interface&gt;&lt;/interface&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/image_inserter_input&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/image_inserter&gt;</td>
</tr>
<tr>
<td><strong>Input source</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;network_input&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;enable_fec_rx&gt;&lt;/enable_fec_rx&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;interface&gt;&lt;/interface&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;password&gt;&lt;/password&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;quad&gt;&lt;/quad&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;udp_igmp_source&gt;&lt;/udp_igmp_source&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;uri&gt;&lt;/uri&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;username&gt;&lt;/username&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/network_input&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;device_input&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;channel&gt;&lt;/channel&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;channel_type&gt;&lt;/channel_type&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;device_id&gt;&lt;/device_id&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;device_name&gt;&lt;/device_name&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;device_number&gt;&lt;/device_number&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;device_type&gt;&lt;/device_type&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;input_format&gt;&lt;/input_format&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;fec_settings&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;udp_igmp_source&gt;&lt;/udp_igmp_source&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;uri&gt;&lt;/uri&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/fec_settings&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;hdmi_settings&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;input_format&gt;&lt;/input_format&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/hdmi_settings&gt;</td>
</tr>
</tbody>
</table>
### Function of code excerpt

**Example code excerpt**

```xml
<sdi_settings>
  <input_format/>
  <scte104_offset/>
</sdi_settings>

<device_input>
  <router_input>
    <input_number/>
    <input_number_end/>
    <quad/>
    <router_ip/>
    <router_type/>
  </router_input>
  <file_input>
    <certificate_file/>
    <interface/>
    <password/>
    <uri/>
    <username/>
  </file_input>
</device_input>

<failover_condition>
  <description/>
  <duration/>
  <order/>
</failover_condition>

<video_selector>
  <color_space/>
  <default_afd/>
  <name/>
  <order/>
  <pid/>
  <program_id/>
</video_selector>

<audio_selector>
  <default_selection/>
  <external_audio_file_input/>
  <infer_external_filename/>
  <language_code/>
  <offset/>
  <order/>
  <pid/>
  <program_selection/>
  <strict_language_selection/>
  <strict_pid_option/>
  <track/>
  <unwrap_smpte337/>
</audio_selector>

<audio_selector_group>
  <audio_selector_name/>
</audio_selector_group>
```
### Implementing Graphic Overlay

You can insert a graphic overlay onto the video in an AWS Elemental Live event using the graphic overlay feature. There are two types of graphic overlay available — static and motion.

#### Topics
- About Graphic Overlay (p. 138)
- Supported Combinations of Features (p. 138)
- Static Graphic Overlay (p. 139)
- Motion Graphic Overlay (p. 150)
- Graphic Overlay Plus Dynamic Content Switching (p. 170)

### Function of code excerpt

<table>
<thead>
<tr>
<th>Source captions</th>
<th>Example code excerpt</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;caption_selector&gt;</td>
<td><code>&lt;caption_selector&gt;</code></td>
</tr>
<tr>
<td><code>&lt;order&gt;</code></td>
<td><code>&lt;order&gt;</code></td>
</tr>
<tr>
<td><code>&lt;source_type&gt;</code></td>
<td><code>&lt;source_type&gt;</code></td>
</tr>
<tr>
<td><code>/source_type&gt;</code></td>
<td><code>/source_type&gt;</code></td>
</tr>
<tr>
<td><code>&lt;language_code&gt;</code></td>
<td><code>&lt;language_code&gt;</code></td>
</tr>
<tr>
<td><code>/language_code&gt;</code></td>
<td><code>/language_code&gt;</code></td>
</tr>
<tr>
<td><code>&lt;embedded_source_settings&gt;</code></td>
<td><code>&lt;embedded_source_settings&gt;</code></td>
</tr>
<tr>
<td><code>&lt;autodetect_scte20&gt;</code></td>
<td><code>&lt;autodetect_scte20&gt;</code></td>
</tr>
<tr>
<td><code>/autodetect_scte20&gt;</code></td>
<td><code>/autodetect_scte20&gt;</code></td>
</tr>
<tr>
<td><code>&lt;source_608_channel_number&gt;</code></td>
<td><code>&lt;source_608_channel_number&gt;</code></td>
</tr>
<tr>
<td><code>/source_608_channel_number&gt;</code></td>
<td><code>/source_608_channel_number&gt;</code></td>
</tr>
<tr>
<td><code>&lt;source_608_track_number&gt;</code></td>
<td><code>&lt;source_608_track_number&gt;</code></td>
</tr>
<tr>
<td><code>/source_608_track_number&gt;</code></td>
<td><code>/source_608_track_number&gt;</code></td>
</tr>
<tr>
<td><code>&lt;upconvert_608_to_708&gt;</code></td>
<td><code>&lt;upconvert_608_to_708&gt;</code></td>
</tr>
<tr>
<td><code>/upconvert_608_to_708&gt;</code></td>
<td><code>/upconvert_608_to_708&gt;</code></td>
</tr>
<tr>
<td><code>&lt;embedded_source_settings&gt;</code></td>
<td><code>&lt;embedded_source_settings&gt;</code></td>
</tr>
<tr>
<td><code>&lt;file_source_settings&gt;</code></td>
<td><code>&lt;file_source_settings&gt;</code></td>
</tr>
<tr>
<td><code>&lt;time_delta&gt;</code></td>
<td><code>&lt;time_delta&gt;</code></td>
</tr>
<tr>
<td><code>/time_delta&gt;</code></td>
<td><code>/time_delta&gt;</code></td>
</tr>
<tr>
<td><code>&lt;upconvert_608_to_708&gt;</code></td>
<td><code>&lt;upconvert_608_to_708&gt;</code></td>
</tr>
<tr>
<td><code>/upconvert_608_to_708&gt;</code></td>
<td><code>/upconvert_608_to_708&gt;</code></td>
</tr>
<tr>
<td><code>&lt;source_file&gt;</code></td>
<td><code>&lt;source_file&gt;</code></td>
</tr>
<tr>
<td><code>&lt;certificate_file&gt;</code></td>
<td><code>&lt;certificate_file&gt;</code></td>
</tr>
<tr>
<td><code>&lt;interface&gt;</code></td>
<td><code>&lt;interface&gt;</code></td>
</tr>
<tr>
<td><code>&lt;password&gt;</code></td>
<td><code>&lt;password&gt;</code></td>
</tr>
<tr>
<td><code>&lt;uri&gt;</code></td>
<td><code>&lt;uri&gt;</code></td>
</tr>
<tr>
<td><code>&lt;username&gt;</code></td>
<td><code>&lt;username&gt;</code></td>
</tr>
<tr>
<td><code>/source_file&gt;</code></td>
<td><code>/source_file&gt;</code></td>
</tr>
<tr>
<td><code>&lt;teletext_source_settings&gt;</code></td>
<td><code>&lt;teletext_source_settings&gt;</code></td>
</tr>
<tr>
<td><code>&lt;page_number&gt;</code></td>
<td><code>&lt;page_number&gt;</code></td>
</tr>
<tr>
<td><code>/page_number&gt;</code></td>
<td><code>/page_number&gt;</code></td>
</tr>
<tr>
<td><code>&lt;dvb_sub_source_settings&gt;</code></td>
<td><code>&lt;dvb_sub_source_settings&gt;</code></td>
</tr>
<tr>
<td><code>&lt;pid&gt;</code></td>
<td><code>&lt;pid&gt;</code></td>
</tr>
<tr>
<td><code>/dvb_sub_source_settings&gt;</code></td>
<td><code>/dvb_sub_source_settings&gt;</code></td>
</tr>
<tr>
<td><code>&lt;scte27_source_settings&gt;</code></td>
<td><code>&lt;scte27_source_settings&gt;</code></td>
</tr>
<tr>
<td><code>&lt;pid&gt;</code></td>
<td><code>&lt;pid&gt;</code></td>
</tr>
<tr>
<td><code>/scte27_source_settings&gt;</code></td>
<td><code>/scte27_source_settings&gt;</code></td>
</tr>
<tr>
<td><code>/caption_selector&gt;</code></td>
<td><code>/caption_selector&gt;</code></td>
</tr>
</tbody>
</table>

### Input clipping

<table>
<thead>
<tr>
<th>Input clipping</th>
<th>Example code excerpt</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;input_clipping&gt;</code></td>
<td><code>&lt;input_clipping&gt;</code></td>
</tr>
<tr>
<td><code>&lt;end_timecode&gt;</code></td>
<td><code>&lt;end_timecode&gt;</code></td>
</tr>
<tr>
<td><code>/end_timecode&gt;</code></td>
<td><code>/end_timecode&gt;</code></td>
</tr>
<tr>
<td><code>&lt;order&gt;</code></td>
<td><code>&lt;order&gt;</code></td>
</tr>
<tr>
<td><code>/order&gt;</code></td>
<td><code>/order&gt;</code></td>
</tr>
<tr>
<td><code>&lt;start_timecode&gt;</code></td>
<td><code>&lt;start_timecode&gt;</code></td>
</tr>
<tr>
<td><code>/start_timecode&gt;</code></td>
<td><code>/start_timecode&gt;</code></td>
</tr>
</tbody>
</table>
About Graphic Overlay

Static Graphic Overlay

The graphic overlay feature lets you insert an image (a BMP, PNG, or TGA file) at a specified time and display it as a static overlay on the underlying video for a specified duration. You can insert up to 8 overlays through the web interface and any number of overlays through the REST interface. The overlays are all independent of each other, which means they can be set up to all appear on the underlying video at the same time (or not), and they can be set up to physically overlap each other (or not). This feature includes fade-in/fade-out capability and opacity.

Motion Graphic Overlay

The motion graphic overlay feature lets you insert an animation (a MOV or SWF file, or a series of PNG files) on the underlying video at a specified time for a specified duration. A typical use case is inserting a spinning corporate logo in a corner of the video.

You can set up only one animation to run over the course of the event. Using the REST interface, you can modify the start time and content of that initial animation in order to achieve the effect of several motion overlays played at different times in the event.

You can insert an animation in the following ways

- MOV Files – Using a .mov file is the most straightforward way to insert an animation onto the video of your event. You provide the file and set up your event to use it.
- Series of PNG Files – You can specify a series of .png files to be inserted one after the other to create an animation.
- SWF Files – In .swf files, you can optionally include Adobe® ActionScript® code that sends a series of commands at a specified time to manipulate the underlying video to provide special effects. You can change the content of the .swf animation during a running AWS Elemental Live event by passing arguments to the ActionScript code via the AWS Elemental Live REST interface. A typical use case is displaying a scoreboard whose content is dynamically refreshed in response to changing scores over the course of a game. SWF files tend to use more system resources than .mov files. Only use .swf if you need a special effect that is available only with that format, such as squeezeback of the underlying video. For information on optimizing your .swf file to run well with AWS Elemental Live, see Motion Graphic Overlay with SWF (p. 201).

Supported Combinations of Features

Combining Static and Motion Overlays

You can insert both a static overlay (or overlays, up to a maximum of 8 overlays) and one motion overlay onto the underlying video.

Combining Different Types of Motion Overlays

You cannot insert more than one type of motion overlay (PNG, MOV, or SWF) at the same time because you can insert only one motion overlay into one event. But you can insert MOV, PNG, and SWF motion overlays onto the same underlying video at different times by using the REST interface to change the
overlay source file and start time. If the first overlay is still running at the start time of the second overlay, the first overlay will stop and the second one immediately starts.

**Combining Graphic Overlays and Dynamic Content Switching**

With AWS Elemental Live, you can combine static overlays with the dynamic content switching feature, described in *Graphic Overlay Plus Dynamic Content Switching* (p. 170). The dynamic content switching feature lets you set up a list of different inputs to be processed by AWS Elemental Live in the same event, one after the other.

**Static Graphic Overlay**

You can insert a static overlay into the event so that it overlays the video in one or more of the video outputs. You can configure the static overlay with a start time and duration. You can insert the static overlay at any position on the video frame, as specified by x/y coordinates. You can configure with an opacity and with fade-in and/or fade-out.

You can insert any number of static overlays in the event.

**Topics**

- Insertion Options and the Effect On Outputs (p. 139)
- Multiple Overlays and Layers (p. 142)
- Combining Overlays and Insertion Options (p. 143)
- Procedure (p. 143)

**Insertion Options and the Effect On Outputs**

**In the Input Section: Insert in Portion of All Outputs**

The static overlay can be inserted in the input section in a specific input.

Result: the portion of the output that is sourced from the given input includes the static overlay.

A typical use case for inserting in this section is if some of the inputs already have a static overlay in the desired location; you do not want to insert another static overlay over this existing static overlay. Therefore, you would insert the static overlay into the inputs that do not have an existing static overlay and omit it from inputs that do have an existing static overlay.

The static overlay will be burned into the video in this input after the input is decoded and before any global processors. This means that the static overlay "sticks" to its input; if the specified start time and duration extend beyond the end of the input, the static overlay will end when the input ends.

If you insert at the Input stage, be very careful with the start time and durations of the static overlays to avoid the static overlay ending abruptly.
In the Global Processors Section: Insert in All Outputs

The static overlay can be inserted in the Global Processors section. Result: The static overlay will be inserted in all outputs.

The static overlay is burned into the video after decoding and input-specific processing and before encoding and creation of individual streams and outputs.
In the Output Section: Insert in One Stream

The static overlay can be inserted in individual streams.
Result: The static overlay is inserted only in the outputs that are associated with those streams. The static overlay will be burned into the video only in the specified streams.

**Overlay Scaling for Each Insertion Option**

The following lists the overlay scaling for each insertion option:

- If you insert the overlay in an input, then when (and if) the underlying video in each output is scaled up or down, the overlay is similarly scaled. The underlying video and overlay maintain their relative ratios. For example, if the overlay covers one quarter of the underlying video in the input, it will cover one quarter of the underlying video in every output.
- If you insert the overlay in the global processors section, then when (and if) the underlying video in each stream assembly is scaled up or down, the overlay is similarly scaled.
- If you insert the overlay in a stream assembly section, then when (and if) the underlying video in each stream assembly is scaled up or down, the overlay is not scaled. The overlay is added only after scaling. Therefore, if the event has two stream assemblies, the final effect of the overlay may be different. The overlay in one stream assembly may take up more room on the underlying video than the overlay in a stream assembly with a different resolution. But the advantage of adding in the stream assembly section is that you can specify different overlays for each assembly, each sized appropriately for that stream's resolution.

**Multiple Overlays and Layers**

You can set up the event to insert more than one static overlay. Overlays are stored in the event in a queue that has a maximum of 8 layers.

This means that you can display up to:

- 8 static overlays if you are using only the web interface to set up the event.
• As many static overlays as you want over the duration of the event if you are using the REST API. See the section called “Step C” (p. 148). A maximum of 8 static overlays can be “queued” at one time (one in each layer).

Combining Overlays and Insertion Options

You can set up the event to insert a static overlay in more than one way. For example, you can insert a static overlay in one layer in the Input section and insert a static overlay in another layer in the Global Processors section.

Examples

• Example 1 – You want to insert a static overlay at a specific time and run it for 10 seconds. You want the static overlay to be placed in the lower right corner of the video frame. You want the static overlay to be 50% opaque and to fade in from nothing to full 50% opacity over 2 seconds, then to fade out to nothing starting 2 seconds before the end of the insertion.

You can implement this use case via the web interface or the REST API.

• Example 2 – You want to insert 2 static overlays so that they both appear on the video frame either at the same time, or with some overlap. You want the display of the static overlays to slightly overlap, so that one static overlay appears in a location and then while that static overlay is still showing, another static overlay appears in a specified location. If the locations overlap either partially or completely, you want to be able to specify which static overlay appears on top.

When you want to insert between 1 and 8 static overlays, you can implement this use case via the web interface or the REST API.

• Example 3 – You want to insert 20 static overlays over the duration of the video. Some of the static overlays repeat, while some are unique. Some of them appear at the same time as others – up to 8 static overlays can appear at the same time.

When you want to insert more than 8 static overlays, you must use the REST API.

• Example 4 – You want the same static overlay to appear repeatedly over the duration of the video, either in the same location each time or in different locations. This use case is simply a variation of use cases 2 or 3 with the same static overlay being used each time.

Procedure

<table>
<thead>
<tr>
<th>Procedural Step</th>
<th>Methods Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Create the event with the desired static overlays.</td>
<td>Use the REST API or the web interface.</td>
</tr>
<tr>
<td>2. Start the event.</td>
<td>Use the REST API or the web interface.</td>
</tr>
<tr>
<td>3. While the event is running, set up more static overlays as desired.</td>
<td>Use the REST API.</td>
</tr>
</tbody>
</table>

You can implement all features described in this section via REST as well as the web interface. This section describes how to use the web interface to insert a graphic overlay. For instructions on doing so via REST, see the section called “Static Overlays via REST API” (p. 171).

The following are valid web interface and REST API combinations:
• Initial setup via web interface and run-time changes via the REST API.
• Initial setup and run-time changes via the REST API.

If you use the web interface to perform the initial setup, note the following restrictions:

• You cannot specify more than 8 static overlays in the event.
• You must set up all the static overlays before the event starts to run; there is no mechanism for changing static overlays at runtime via the web interface.

Topics
• Step A: Prepare the Overlay Asset (p. 144)
• Step B: Initial Setup (p. 144)
• Step C: Manage Overlays on a Running Event (p. 148)

Step A: Prepare the Overlay Asset

1. Create a file with the following characteristics:
   • File type: A BMP, PNG, or TGA file.
   • Aspect ratio: The overlay can have any aspect ratio. It does not have to match the aspect ratio of the underlying video.
   • Size, in pixels: The overlay can be any size up to the same size as the underlying video. The overlay cannot be positioned so that part of the overlay runs beyond the right edge or bottom edge of the underlying video.
   • If you set up an overlay so that it is too big or it overruns an edge, if AWS Elemental Live can identify this error at event creation time, an error message will appear then.
   • If AWS Elemental Live cannot identify the error in advance, an error message will appear while the event is running. The event will not stop but the insertion request will fail.

2. Place the prepared file in a location accessible to the AWS Elemental Live node. You can specify the location in one of the following ways:
   • Local to the AWS Elemental Live appliance. E.g. /data/assets/overlay.mov
   • A remote server via a mount point. E.g. /data/mnt/assets/overlay.mov
   • An S3 bucket, using SSL. E.g. s3ssl://company.test/sample_bucket/overlay.mov
   • An S3 bucket, without SSL. E.g. s3://company.test/sample_bucket/overlay.mov

   Use sse=true to enable S3 Server Side Encryption (SSE) and rrs=true to enable Reduced Redundancy Storage (RRS). Default values for RRS and SSE are false.

   Example: s3://<hostname>/sample_bucket/encrypted?rrs=true&sse=true

3. Make a note of the location.

Step B: Initial Setup

Create or modify the event as follows:

1. Determine the location or locations in the event where the static overlay should be inserted, then display the appropriate section:
   • Input section: In the desired input or inputs, click Advanced. More fields appear. In the Image Inserter section, click On. More fields appear; see the table below.
• Global Processors section: In the Global Processors section, go to the Image Inserter field and click On. More fields appear; see the table below.

• Output section: In the desired output or outputs, determine the stream this output is associated with. In the corresponding Stream section, click Advanced. More fields appear; see the table below.

For all locations, the following fields appear. (Note that the following image is from the Global Processors section, but the fields are the same in all sections.)

2. Complete the fields as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Image Location    | The location and filename of the PNG or BMP image file.  
For file requirement details, information about where to store this file, and how to specify its location, see the section called “Step A” (p. 144).  
For S3, use sse=true to enable S3 Server Side Encryption (SSE) and rrs=true to enable Reduced Redundancy Storage (RRS). Default values for RRS and SSE are false. |
| Username Password | If access to your local or mounted directory requires a username and password, click the lock icon next to the Image Location field to show the Username and Password fields.  
For S3, Enter the Access Key ID in the username field. Enter the Secret Access Key in the password field. |
| Layer             | A number from 0 to 7 for the Z order of the static overlay. “Z order” means that static overlays with higher values of layer will be inserted on top of static overlays with lower values of layer.  
Default is 0. |
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>Placement of the left edge of the motion overlay relative to the left edge of the video frame, in pixels. 0 is the left edge of the frame. Take note of the width of the motion overlay and make sure that the position of the left edge of the motion overlay does not cause the right edge to be cropped.</td>
</tr>
<tr>
<td>Top</td>
<td>Placement of the top edge of the motion overlay relative to the top edge of the video frame, in pixels. 0 is the top edge of the frame. Default is 0. Take note of the height of the motion overlay and make sure that the position of the top edge of the motion overlay does not cause the bottom edge to be cropped.</td>
</tr>
<tr>
<td>Opacity</td>
<td>The opacity of the static overlay, as a number from 0 to 100. 0 is transparent. 100 is fully opaque. Default is 50.</td>
</tr>
<tr>
<td>Width</td>
<td>The width of the static overlay when inserted in the video, in pixels. Leave blank to use the native width of the static overlay. The original static overlay will be scaled up or down, to the specified width.</td>
</tr>
<tr>
<td>Height</td>
<td>The height of the static overlay when inserted in the video, in pixels. Leave blank to use the native height of the static overlay. The original static overlay will be scaled up or down, to the specified height.</td>
</tr>
<tr>
<td>Start Time</td>
<td>The start time for the overlay. Specify the start time in one of the formats listed at the section called “Start Time Formats” (p. 147).</td>
</tr>
<tr>
<td>Duration</td>
<td>The amount of time, in milliseconds, for the overlay to remain on the video. If this field is left blank, the static overlay will remain on the video as follows: • In the Input section: Until this input ends. • In the Global Processor section: Until the event ends. • In the Output section: Until the event ends. The total running time of the static overlay is Fade in + Duration + Fade out.</td>
</tr>
</tbody>
</table>
Field | Description
--- | ---
Fade In | The duration, in milliseconds, for the static overlay fade-in. This time is inserted before the start time of the static overlay.
Fade Out | This field is valid only if the Duration field is completed. The duration, in milliseconds, for the static overlay fade-out. This time is added to the static overlay duration.
Enable Rest Control | Check this field only if you plan to manage motion overlays via the REST API, after this initial setup via the web interface. Typically, you will want this tag to be true.

3. If desired, click Add Image and enter the information for another static overlay, up to a maximum of 8 static overlays.
   • Assign a unique Layer number to each static overlay. The layers do not have to appear in any particular order on the screen, but each number must be used once only.
   • The static overlay Start Time and Duration can be set so that any static overlay overlaps the display time of any other static overlay.
   • The Left/Top fields can be set so that any static overlay physically overlaps any other static overlay, as much as you want; the static overlays are displayed according to their Layer value.

Note: If you are using input switching to provide input redundancy (with or without the hot-hot backup mode), then make sure you insert the static overlay in both input pairs.

Start Time Formats

**Option 1:** Timecode format (HH:MM:SS:FF). The overlay start is determined by comparing the specified start to the appropriate timecode.

- If the overlay is specified in the Input section: The start time for the static overlay will be compared to the input timecode (the timecode for a given input). The source for the input timecode is specified separately for each input (Input > Timecode Source field). The input timecode is calculated as follows:
  - If Timecode Source is Embedded: The timecode associated with each frame is extracted from the timecode carried with the input media. Note that each input will have its own timecode and the timecode may not align well from one input to another.
  - If Timecode Source field is Start at 0: The timecode of the first frame of the input is 00:00:00:00 and the timecode counts up with each successive frame. The timecode starts over with each input.
  - If Timecode Source field is System Clock or Local System Clock (AWS Elemental Live only): The timecode of each frame in the input is the system time at which the frame is decoded.
- If the overlay is specified in the Global Processor section: The overlay start is compared to the output timecode (which is shared by all outputs). The source for the output timecode is specified for the entire event, in the Timecode Config > Source field. The output timecode is calculated as follows:
  - If Source is Embedded: The timecode is extracted from the timecode carried with the input media. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.
  - If Source is Start at 0: The output timecode for the first frame is 00:00:00:00 and then the output timecode counts up with each successive frame in the entire output.
• If Source is System Clock or Local System Clock (AWS Elemental Live only): The output timecode for the first frame is the system time at which the frame is decoded. Then the output timecode counts up with each successive frame in the entire output.
• If Source is Specified Start: The output timecode for the first frame is the time you specified when you selected this option as the timecode source. Then the output timecode counts up with each successive frame in the entire output.
• If Source is External Reference Connector (AWS Elemental Live only): The timecode is extracted from external LTC source. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.
• If the static overlay is specified in the Output section: The start time for the static overlay is calculated in the same way as a static overlay in the Global Processor section.

Option 2: ISO 8601 UTC time with no dashes or colons. For example, 20160102T030405.678Z. In this case, the start time for every overlay (regardless of whether it is defined in the input, the global processor or the output) will be the UTC time.

Step C: Manage Overlays on a Running Event

Once the event has started, you can work with static overlays only via the REST API. You cannot work with static overlays on a running event via the web interface.

Change the static overlay or overlays on a running event:

1. If you did not set the <enable_rest> element to true when you created the event, modify the event (PUT Event) and set this value. For the location of this element, see the section called "Modify Static Overlay" (p. 179).
2. Send the Modify Static Overlay command (see the section called "Static Overlays via REST API" (p. 171)) to make the desired change to the static overlays in the event.

Runtime REST Commands Change the Event XML

When you send REST commands during an event, the event XML is permanently changed. Any data sent via the REST call goes into the XML.

For example, you might put a scoreboard overlay on your event during a sporting event. If you do not send a REST call to deactivate the overlay once the game has ended, the scoreboard will appear again at the same time the next day.

Therefore, make sure to do one of the following:

• If you plan to run the event (event A) again with different video content but with the same graphic overlays, make sure to export your XML for re-use before starting the event. Then create a new event (event B) using the exported XML. Do not start event A again.
• If you are running a 24/7 channel and you do not want your overlay to recur, remember to send a REST command to set <activate> to false once the overlay has run. This will delete the entire <insertable_images> element from the event XML.

You could also specify an absolute start time for each overlay by using the ISO 8601 UTC time format to specify an absolute time and date for the overlay. The overlays will not run again the next day.

Options for Insertion - Which Outputs are Affected

You can insert static overlays in one of the following places in the running event. These places are the same as the places when inserting in a new event or modifying a non-running event.

• In an individual input. The input must be currently running or be in the future.
If you include a start time in the XML body, that start time must fall within the scope of the specified input. It must correspond to a time that is valid for that input. For example, if the input runs from 1:00 p.m. to 2:00 p.m. by the clock, the start time must correspond to a clock time between 1:00 p.m. and 2:00 p.m., otherwise the insertion will be ignored. The start time can be in the past, so long as it is within the scope of the input; in this case, the overlay will be inserted immediately.

- In all outputs.

If you include a start time in the XML body, the overlay will be inserted at that at start time. If that start time is in the past, the overlay will be inserted immediately.

- In the outputs associated with one stream assembly.

If you include a start time in the XML body, the overlay will be inserted at that at start time. If that start time is in the past, the overlay will be inserted immediately.

**Types of Changes**

**Add an Overlay in a Layer**

You can add an overlay in a layer. For example, if you did not fill all 8 layers when creating the event, you can add more static overlays, up to a total of 8 for the event. Or if you already deleted a layer (as described below), you can fill it with a new static overlay.

You must enter a Modify Static Overlay command ([the section called “Create or Modify a Non-Running Event” (p. 172)](#)) and include the following tags in the XML body:

- layer: The (unused) layer where you want to add the static overlay.
- activate: Set to true. Note that this tag is not part of the XML body for creating the event.
- Other tags: Set all other tags as desired to specify the content and its characteristics, start time and duration.

**Modify an Existing Overlay**

You can modify the existing content in a layer. You can do the following:

- Change a static overlay that has not yet run.
- Change a static overlay that is currently running. The static overlay will change in mid-stream.
- Change a static overlay that has run in order to re-use the layer.

You can change the static overlay's start time or duration. Or you can change its position. Or you can change the actual overlay that runs.

You must enter a Modify Static Overlay command ([the section called “Create or Modify a Non-Running Event” (p. 172)](#)) and include the following tags in the XML body:

- layer: The layer whose static overlay you want to modify.
- activate: Set to true. Note that this tag is not part of the XML body for creating the event.
- Other tags: Set all other tags as desired to specify the content and its characteristics, start time and duration. Only the tags you specify will be changed.

**Delete an Overlay**

You can delete the existing content in a layer. If the static overlay has not yet run, it will not run. If the static overlay is currently running, it will be removed. If the static overlay has already run, there is not really any need to delete the content.
You must enter a Modify Static Overlay command (the section called “Create or Modify a Non-Running Event” (p. 172)) and include the following tags in the XML body:

- layer: The layer to delete.
- activate: Set to false. Note that this tag is not part of the XML body for creating the event.

**Motion Graphic Overlay**

You can insert a motion overlay into the event so that it overlays the video in all the video outputs. You can insert only one motion overlay into the event, but you can use the REST interface during the event to cause the overlay to run repeatedly, either with the same content or new content. Only one motion asset can be run at a time in this way.

**Supported File Types**

You can use a .swf file, a .mov file, or a set of ordered still .png files as assets for motion image overlays. For .swf and .mov files, the frame rate of the overlay asset must match the frame rate of the underlying video. For sets of .png files, you specify the frame rate when you set up the overlay.

**Options for Insertion**

The motion overlay is a global processor, which means that it will be inserted in all outputs. The overlay will be burned into the video after decoding and input-specific processing and before encoding and creation of individual streams and outputs. Note that unlike static overlays, a motion overlay can only be set up as a global processor. It cannot be specified in the input (so that it applies only to one video input) and it cannot be specified in the output (so that it applies only to one output stream).

**Examples**

- **Example 1:** “Coming up” Motion Overlay – You want to insert a PNG asset (a series of PNG files that will run as a 10 second motion overlay) 59 minutes into the runtime of the event. The motion overlay may be a “coming up” motion overlay. You want the motion overlay to be placed in the lower right corner of the video frame. You can implement this use case via the web interface or the REST API.

- **Example 2:** Animated Corporate Logo – You want to insert a PNG asset (a series of PNG files that will run as a 10 second motion overlay) and run it every 5 minutes starting 20 minutes into the runtime of the event. You specify the single asset in the event and specify the first runtime. You start the event then, after the motion overlay has run once, you send a REST API command to modify the start time of the asset to a new time. You repeat this command (each time with a new start time) as many times as you want.

- **Example 3:** Combo – You want to insert a motion PNG asset (a series of PNG files that will run as a 10 second motion overlay) and run it every 5 minutes starting 20 minutes into the runtime of the event. You specify the single asset in the event and specify the first runtime. You start the event then, after the motion overlay has run once, you use the REST API to modify the start time of the asset to a new time. You repeat this command (each time with a new start time) as many times as you want.

Two minutes before the end of the event, you want a different PNG asset to show (a “coming up” motion overlay). You send a REST API command to change the start time and the content.

**Topics**

- How to Insert a Motion Overlay with Quicktime MOV (p. 151)
- How to Insert a Motion Overlay with a Set of PNG Files (p. 156)
- How to Insert a Motion Overlay with a SWF File (p. 163)
How to Insert a Motion Overlay with Quicktime MOV

The motion overlay can be configured with a start time and to run only once or to loop until you optionally clear the insertion. The motion overlay can overlay the video at any position, as specified by number of pixels offset from the upper left corner.

Web Interface and REST API

<table>
<thead>
<tr>
<th>Procedural Step</th>
<th>Methods Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Prepare the MOV file.</td>
<td>Use a third-party process.</td>
</tr>
<tr>
<td>2. Create the event with the desired motion overlays. As part of the creation,</td>
<td>Use the REST API or the web interface</td>
</tr>
<tr>
<td>make sure that the live_event XML file includes a motion_image tag. Also</td>
<td></td>
</tr>
<tr>
<td>include an action_time tag and a loop tag (set to true).</td>
<td></td>
</tr>
<tr>
<td>3. Start the event and wait for the motion overlay to start playing.</td>
<td>Use the REST API or the web interface</td>
</tr>
<tr>
<td>4. Set up more motion overlays as desired</td>
<td>Use the REST API</td>
</tr>
</tbody>
</table>

The following are valid web interface and REST API combinations:

- Initial setup via web interface and run-time changes via the REST
- Initial setup and run-time changes via the REST API.

If you use the web interface to perform the initial setup, note the following restrictions:

- There is no mechanism for changing motion overlays at runtime via the web interface.

Step A: Prepare the MOV Asset

1. Create a File – Use a third-party process to create a MOV file encoded with Apple Quicktime Run Length Encoding. Other codecs in the MOV container are not supported. Take care with the following settings:
   - Aspect ratio: The motion overlay can have any aspect ratio. It does not have to match the aspect ratio of the video output.
   - Size: The motion overlay can be any size, in pixels, up to the size of the underlying video. The motion overlay must be prepared in the desired size; there is no way to resize it when setting it up in the event.
   - Position: The overlay cannot be positioned so that part of the overlay runs beyond the right edge or bottom edge of the underlying video.
     - If you set up a motion overlay so that it is too big or it overruns and AWS Elemental Live can identify this error at event creation time, then an error message will appear.
     - If AWS Elemental Live cannot identify the error in advance, an error message will appear while the event is running. The event will not stop but the insertion request will fail.

2. Place the File – Place the MOV file in a location accessible to the AWS Elemental Live: on a local directory, on a remote filesystem accessible via mount point, or in an S3 bucket.
You can specify the location in one of the following ways:

- Local to the AWS Elemental Live appliance. E.g. /data/assets/overlay.mov
- A remote server via a mount point. E.g. /data/mnt/assets/overlay.mov
- An S3 bucket, using SSL. E.g. s3ssl://company.test/sample_bucket/overlay.mov
- An S3 bucket, without SSL. E.g. s3://company.test/sample_bucket/overlay.mov

**Step B: Set Up the Event**

Using the web interface, create or modify a non-running event as follows:

1. In the Global Processors section, go to the Image Inserter field and click On. More fields appear; see the table below.

2. Complete the fields as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Insertion Mode</strong></td>
<td>Choose MOV.</td>
</tr>
<tr>
<td><strong>Input</strong></td>
<td>The location and filename of the MOV file.</td>
</tr>
<tr>
<td></td>
<td>For S3, use sse=true to enable S3 Server Side Encryption (SSE) and rrs=true to enable Reduced Redundancy Storage (RRS). Default values for RRS and SSE are false.</td>
</tr>
<tr>
<td><strong>Username Password</strong></td>
<td>If access to your local or mounted directory requires a username and password, click the lock icon next to the Input field to show the Username and Password fields.</td>
</tr>
<tr>
<td></td>
<td>For S3, Enter the Access Key ID in the username field. Enter the Secret Access Key in the password field.</td>
</tr>
<tr>
<td><strong>Left</strong></td>
<td>Placement of the left edge of the motion overlay relative to the left edge of the video frame, in pixels. 0 is the left edge of the frame.</td>
</tr>
<tr>
<td></td>
<td>Take note of the width of the motion overlay and make sure that the position of the left edge of the motion overlay does not cause the right edge to be cropped.</td>
</tr>
</tbody>
</table>
# Motion Graphic Overlay

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Top**       | Placement of the top edge of the motion overlay relative to the top edge of the video frame, in pixels. 0 is the top edge of the frame. Default 0.  
Take note of the height of the motion overlay and make sure that the position of the top edge of the motion overlay does not cause the bottom edge to be cropped. |
| **ActionTime**| The start time for the overlay. Specify the start time in one of the formats discussed in detail below this table. |
| **Loop Input**| • Check to loop the motion overlay indefinitely.  
The motion overlay will run until the event ends. To stop the overlay earlier, see the section on modifying overlays on a running event, in the section called "Motion Overlays via REST API" (p. 180).  
• Clear checkbox to run the motion overlay only once. |
| **Full Frame**| Expand the overlay to fit the video frame. In this case, make sure Left and Top are set to 0.  
If this field is checked and the motion overlay has a different aspect ratio to the underlying video, the motion overlay will be scaled until either:  
• The motion overlay fits in the length. The overlay will then be positioned with equal space on the left and right.  
• The motion overlay fits in the width. The overlay will then be positioned with equal space above and below.  
Note that the “Stretch to output” field in the Stream section does not affect the motion overlay; even if the video is stretched, the overlay is not stretched. |
| **Active**    | Always check this box when initially setting up the motion overlay.  
After the initial setup, the value of this tag can be changed via the REST API to manage the content and behavior of the motion overlay. |
| **Enable Rest Control** | Check this field only if you plan to manage motion overlays via the REST API, after this initial setup via the web interface. Typically, you will want this tag to be true. |
**Action Time Formats**

**Option 1:** Timecode format (HH:MM:SS:FF). If Timecode Source field is System Clock or Local System Clock (AWS Elemental Live only): The timecode of each frame in the input is the system time at which the frame is decoded.

The overlay start is compared to the output timecode (which is shared by all outputs). The source for the output timecode is specified for the entire event, in the Timecode Config > Source field. The output timecode is calculated as follows:

- If Source is Embedded: The timecode is extracted from the timecode carried with the input media. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.
- If Source is Start at 0: The output timecode for the first frame is 00:00:00:00 and then the output timecode counts up with each successive frame in the entire output.
- If Source is System Clock or Local System Clock (AWS Elemental Live only): The output timecode for the first frame is the system time at which the frame is decoded. Then the output timecode counts up with each successive frame in the entire output.
- If Source is Specified Start: The output timecode for the first frame is the time you specified when you selected this option as the timecode source. Then the output timecode counts up with each successive frame in the entire output.
- If Source is External Reference Connector (AWS Elemental Live only): The timecode is extracted from external LTC source. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.

**Option 2:** ISO 8601 UTC time with no dashes or colons. For example, 20160102T030405.678Z. In this case, the start time for every overlay will be the UTC time.

**Option 3:** You can only use this while adding or modifying an overlay in a running event. Set this tag to an empty string to set the start time to “now”. With this option, the start time is never exact. You cannot use this option when creating an event or modifying a non-running event.

**Step C: Manage the Overlay on a Running Event**

After the event starts, you can work with motion overlays only via the REST API. You cannot work with motion overlays on a running event via the web interface.

The motion overlay feature lets you specify only one motion overlay, but using the REST API, you can re-use it (start it again and again), change its content each time it is run, and change its attributes (for example, size) each time it runs.

1. If you did not set the enable_rest tag to true when you created the event, modify the event (PUT Live Event) and set this value to true. Do this before the event is running. See the section called “Motion Overlays via REST API” (p. 180) for the location of this tag.
2. Send the Modify Overlay on Running Event command to make the desired change to the overlays in the event. See the following table for information on the types of changes you can make to a running event via the REST API:

<table>
<thead>
<tr>
<th>Current State of Overlay</th>
<th>Desired Action</th>
<th>What to Do</th>
<th>Tags to Include in Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not running</td>
<td>Enable overlay control</td>
<td>Enter a Modify Overlay</td>
<td>enable_rest</td>
</tr>
<tr>
<td></td>
<td>via the REST API</td>
<td>on Running Event</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Description</td>
<td>Command</td>
<td>Rules</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>Not running</td>
<td>Run the motion overlay immediately</td>
<td>Enter a Modify Overlay on Running Event command to set the <code>&lt;active&gt;</code> tag to true and set the <code>&lt;action_time&gt;</code> tag to empty.</td>
<td>active action_time</td>
</tr>
<tr>
<td>Not running</td>
<td>Run the motion overlay again at a specified time</td>
<td>Enter a Modify Overlay on Running Event command to set the <code>&lt;active&gt;</code> tag to true and set the <code>&lt;action_time&gt;</code> tag to the desired start time.</td>
<td>active action_time</td>
</tr>
<tr>
<td>Not running</td>
<td>Run a different motion overlay of the same file type.</td>
<td>Enter a Modify Overlay on Running Event command to change the input to point to a different motion graphic asset. See the section called &quot;Motion Overlays via REST API&quot; (p. 180). Set the <code>&lt;active&gt;</code> tag to true and set the <code>&lt;action_time&gt;</code> tag to the time you want the start the new motion overlay. Set the <code>&lt;loop&gt;</code> tag to true or false.</td>
<td>All tags that apply to the current file type.</td>
</tr>
<tr>
<td>Not running</td>
<td>Run a different motion overlay of a different file type.</td>
<td>Enter a Modify Overlay on Running Event command and set the input tag to point to a different type of asset, as described in the appropriate section of this chapter. Set the <code>&lt;active&gt;</code> tag to true and set the <code>&lt;action_time&gt;</code> tag to the time you want the start of the new motion overlay. Set the <code>&lt;loop&gt;</code> tag to true or false.</td>
<td>All tags that apply to the file type you want to run.</td>
</tr>
</tbody>
</table>

Do not include any other tags!
Running | Stop the running motion overlay | Just before you want the motion overlay to stop, enter a Modify Overlay on Running Event command to set the active tag to false. The motion overlay will immediately be stopped.

Running | Start the motion overlay | To start the motion overlay again, enter a Modify Overlay on Running Event command to set the <active> tag to true and the <action_time> tag set to the time you want the overlay to begin.

Running | Change to a different motion overlay | Enter a Modify Overlay on Running Event command to change the input to a different file. All tags, not just the ones you want to change. If you exclude a tag, the default value will apply.

**Runtime REST Commands Change the Event XML**

When you send REST commands during an event, the event XML is permanently changed. Any data sent via the REST call goes into the XML.

For example, you might put a scoreboard overlay on your event during a sporting event. If you do not send a REST call to deactivate the overlay once the game has ended, the scoreboard will appear again at the same time the next day.

Therefore, make sure to do one of the following:

- If you plan to run the event (event A) again with different video content but with the same graphic overlays, make sure to export your XML for re-use before starting the event. Then create a new event (event B) using the exported XML. Do not start event A again.
- If you are running a 24/7 channel and you do not want your overlay to recur, remember to send a REST command to set <activate> to false once the overlay has run. This will delete the entire <insertable_images> element from the event XML.

You could also specify an absolute start time for each overlay by using the ISO 8601 UTC time format to specify an absolute time and date for the overlay. The overlays will not run again the next day.

**How to Insert a Motion Overlay with a Set of PNG Files**

You use a third-party process to convert a motion overlay to a series of PNG files. Then place these files in a directory AWS Elemental Live can access, either locally, through a mount point, or from an S3 bucket. Then set up your event to insert these files at a given framerate (for example, 30 frames/second).
While the underlying video is being processed, the PNG files are “played” at the specified framerate and are burned onto the underlying video.

The motion overlay can be configured with a start time and to run only once or to loop until you optionally clear the insertion. The animation can overlay the video at any position, as specified by number of pixels offset from the upper left corner.

Web Interface and REST API

<table>
<thead>
<tr>
<th>Procedural Step</th>
<th>Methods Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Prepare the MOV file.</td>
</tr>
<tr>
<td></td>
<td>Use a third-party process.</td>
</tr>
<tr>
<td>2.</td>
<td>Create the event with the desired motion overlays. As part of the creation, make sure that the live_event XML file includes a motion_image tag. Also include an action_time tag and a loop tag.</td>
</tr>
<tr>
<td></td>
<td>Use the REST API or the web interface.</td>
</tr>
<tr>
<td>3.</td>
<td>Start the event and wait for the motion overlay to start playing.</td>
</tr>
<tr>
<td></td>
<td>Use the REST API or the web interface.</td>
</tr>
<tr>
<td>4.</td>
<td>Set up more motion overlays as desired</td>
</tr>
<tr>
<td></td>
<td>Use the REST API.</td>
</tr>
</tbody>
</table>

The following are valid web interface and REST API combinations:

- Initial setup via web interface and run-time changes via the REST API.
- Initial setup and run-time changes via the REST API.

If you use the web interface to perform the initial setup, note the following restrictions:

- There is no mechanism for changing motion overlays at runtime via the web interface.

Step A: Prepare the PNG Asset

1. Create a File – Use a third-party process to convert an animation asset to a series of PNG files. Take care with these aspects of the conversion:

   - File count: When you insert the files into the video, you will specify the framerate for the motion overlay. Therefore, make sure that the conversion results in a file count that aligns with the intended framerate. (The framerate of the motion overlay does not necessarily have to match the framerate of the underlying video.) For example, if the motion overlay will run for 10 seconds at 30 frames/second, make sure the conversion produces 300 files. If the file count and framerate do not align, the quality of the animation may suffer.
   - The PNG files must contain RGB and one alpha channel. The alpha channel will be used for per-pixel blending.
   - File names: Make sure the filenames of the converted files include a sequential number. There can be any number of digits in the numerical part of the filename, as long as it is the same for each file. For example, 001 to 200 but not 01 to 200. Note that the sequential numbering does not have to start at zero.
   - Aspect ratio: The motion overlay can have any aspect ratio. It does not have to match the aspect ratio of the video output.
• Size: The motion overlay can be any size, in pixels, up to the size of the underlying video. The motion overlay must be prepared in the desired size; there is no way to resize it when setting it up in the event.

• Position: The overlay cannot be positioned so that part of the overlay runs beyond the right edge or bottom edge of the underlying video.

• If you set up a motion overlay so that it is too big or it overrun and AWS Elemental Live can identify this error at event creation time, then an error message will appear.

• If AWS Elemental Live cannot identify the error in advance, an error message will appear while the event is running. The event will not stop but the insertion request will fail.

2. Place the File – Place the converted file in a location accessible to AWS Elemental Live: on a local directory, on a remote filesystem accessible via mount point, or in an S3 bucket. Choose a location as described in the table below, then note the location for setting up the overlay in the event.

You can specify the location in one of the following ways:

• Local to the AWS Elemental Live appliance. E.g. /data/assets/overlay_001.png
• A remote server via a mount point. E.g. /data/mnt/assets/overlay_001.png
• An S3 bucket, using SSL. E.g. s3ssl://company.test/sample_bucket/overlay_001.png
• An S3 bucket, without SSL. E.g. s3://company.test/sample_bucket/overlay_001.png

**Step B: Set up the Event**

Using the web interface, create or modify a non-running event as follows:

1. In the Global Processors section, go to the Image Inserter field and click On. More fields appear; see the table below.

2. Complete the fields as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion Mode</td>
<td>Choose PNG.</td>
</tr>
<tr>
<td>Input</td>
<td>The path and filename of the PNG files. Provide the path and filename of the first PNG file in the series. All files in the series must have the same number of digits in the numerical part of the filename. For example, if the files are stored on /mnt/storage/motion_logos/ and the files are named logo_hi_001 to logo_hi_357, enter “/mnt/storage/motion_logos/logo_hi_001”.</td>
</tr>
</tbody>
</table>
When using S3, you can optionally append the path as follows:

- Use `sse=true` to enable S3 Server Side Encryption (SSE).
- Use `rrs=true` to enable Reduced Redundancy Storage (RRS).

Default values for RRS and SSE are false.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username Password</td>
<td>If access to your local or mounted directory requires a username and password, click the lock icon next to the Input field to show the Username and Password fields. For S3, Enter the Access Key ID in the username field. Enter the Secret Access Key in the password field.</td>
</tr>
<tr>
<td>Left</td>
<td>Placement of the left edge of the motion overlay relative to the left edge of the video frame, in pixels. 0 is the left edge of the frame. Take note of the width of the motion overlay and make sure that the position of the left edge of the motion overlay does not cause the right edge to be cropped.</td>
</tr>
<tr>
<td>Top</td>
<td>Placement of the top edge of the motion overlay relative to the top edge of the video frame, in pixels. 0 is the top edge of the frame. Default 0. Take note of the height of the motion overlay and make sure that the position of the top edge of the motion overlay does not cause the bottom edge to be cropped.</td>
</tr>
<tr>
<td>ActionTime</td>
<td>The start time for the overlay. Specify the start time in one of the formats discussed in detail below this table.</td>
</tr>
<tr>
<td>Numerator Denominator</td>
<td>The framerate as a numerator over a denominator. For example, 29.97 fps is a numerator of 30000 and a denominator of 1001. Enter numbers that give a framerate ratio between 1 and 120.</td>
</tr>
</tbody>
</table>
### Field | Description
---|---
Loop Input | Check the checkbox to loop the motion overlay indefinitely. The motion overlay will run until the event ends. To stop the overlay earlier, see the section on modifying overlays on a running event, in the section called “Motion Overlays via REST API” (p. 180).
| Clear the checkbox to run the motion overlay only once.

Full Frame | Expand the overlay to fit the video frame. In this case, make sure Left and Top are set to 0.
| If this field is checked and the motion overlay has a different aspect ratio to the underlying video, the motion overlay will be scaled until either:
| • The motion overlay fits in the length. The overlay will then be positioned with equal space on the left and right.
| • The motion overlay fits in the width. The overlay will then be positioned with equal space above and below.
| Note that the “Stretch to output” field in the Stream section does not affect the motion overlay; even if the video is stretched, the overlay is not stretched.

Active | Always check this box when initially setting up the motion overlay.
| After the initial setup, the value of this tag can be changed via the REST API to manage the content and behavior of the motion overlay.

Enable Rest Control | Check this field only if you plan to manage motion overlays via the REST API, after this initial setup via the web interface. Typically, you will want this tag to be true.

### Action Time Formats

**Option 1:** Timecode format (HH:MM:SS:FF). If Timecode Source field is System Clock or Local System Clock (AWS Elemental Live only): The timecode of each frame in the input is the system time at which the frame is decoded.

The overlay start is compared to the output timecode (which is shared by all outputs). The source for the output timecode is specified for the entire event, in the Timecode Config > Source field. The output timecode is calculated as follows:

- If Source is Embedded: The timecode is extracted from the timecode carried with the input media. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.
• If Source is Start at 0: The output timecode for the first frame is 00:00:00:00 and then the output timecode counts up with each successive frame in the entire output.

• If Source is System Clock or Local System Clock (AWS Elemental Live only): The output timecode for the first frame is the system time at which the frame is decoded. Then the output timecode counts up with each successive frame in the entire output.

• If Source is Specified Start: The output timecode for the first frame is the time you specified when you selected this option as the timecode source. Then the output timecode counts up with each successive frame in the entire output.

• If Source is External Reference Connector (AWS Elemental Live only): The timecode is extracted from external LTC source. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.

Option 2: ISO 8601 UTC time with no dashes or colons. For example, 20160102T030405.678Z. In this case, the start time for every overlay will be the UTC time.

Option 3: You can only use this while adding or modifying an overlay in a running event. Set this tag to an empty string to set the start time to “now”. With this option, the start time is never exact. You cannot use this option when creating an event or modifying a non-running event.

Step C: Manage the Overlay on a Running Event

After the event starts, you can work with motion overlays only via the REST API. You cannot work with motion overlays on a running event via the web interface.

The motion overlay feature lets you specify only one motion overlay, but using the REST API, you can re-use it (start it again and again), change its content each time it is run, and change its attributes (for example, size) each time it runs.

1. If you did not set the enable_rest tag to true when you created the event, modify the event (PUT Live Event) and set this value to true. Do this before the event is running. See the section called “Motion Overlays via REST API” (p. 180) for the location of this tag.

2. Send the Modify Overlay on Running Event command to make the desired change to the overlays in the event. See the following table for information on the types of changes you can make to a running event via the REST API:

<table>
<thead>
<tr>
<th>Current State of Overlay</th>
<th>Desired Action</th>
<th>What to Do</th>
<th>Tags to Include in Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not running</td>
<td>Enable overlay control via the REST API</td>
<td>Enter a Modify Overlay on Running Event command to set the <code>&lt;enable_rest&gt;</code> tag to true, in case you did not set it to true when you created the event.</td>
<td>enable_rest Do not include any other tags!</td>
</tr>
<tr>
<td>Not running</td>
<td>Run the motion overlay immediately</td>
<td>Enter a Modify Overlay on Running Event command to set the <code>active_tag</code> tag to true and set the <code>&lt;action_time&gt;</code> tag to empty.</td>
<td>active action_time</td>
</tr>
</tbody>
</table>

Version 2.17
<table>
<thead>
<tr>
<th>State</th>
<th>Action Description</th>
<th>Command Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not running</td>
<td>Run the motion overlay again at a specified time</td>
<td>Enter a Modify Overlay on Running Event command to set the <code>&lt;active&gt;</code> tag to true and set the <code>&lt;action_time&gt;</code> tag to the desired start time.</td>
</tr>
<tr>
<td>Not running</td>
<td>Run a different motion overlay of the same file type.</td>
<td>Enter a Modify Overlay on Running Event command to change the input to point to a different motion graphic asset. See the section called &quot;Motion Overlays via REST API&quot; (p. 180). Set the <code>&lt;active&gt;</code> tag to true and set the <code>&lt;action_time&gt;</code> tag to the time you want the start the new motion overlay. Set the <code>&lt;loop&gt;</code> tag to true or false.</td>
</tr>
<tr>
<td>Not running</td>
<td>Run a different motion overlay of a different file type.</td>
<td>Enter a Modify Overlay on Running Event command and set the input tag to point to a different type of asset, as described in the appropriate section of this chapter. Set the <code>&lt;active&gt;</code> tag to true and set the <code>&lt;action_time&gt;</code> tag to the time you want the start of the new motion overlay. Set the <code>&lt;loop&gt;</code> tag to true or false.</td>
</tr>
<tr>
<td>Running</td>
<td>Stop the running motion overlay</td>
<td>Just before you want the motion overlay to stop, enter a Modify Overlay on Running Event command to set the <code>&lt;active&gt;</code> tag to false. The motion overlay will immediately be stopped.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Action Description</th>
<th>Command Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not running</td>
<td>Run the motion overlay again at a specified time</td>
<td>Enter a Modify Overlay on Running Event command to set the <code>&lt;active&gt;</code> tag to true and set the <code>&lt;action_time&gt;</code> tag to the desired start time.</td>
</tr>
<tr>
<td>Not running</td>
<td>Run a different motion overlay of the same file type.</td>
<td>Enter a Modify Overlay on Running Event command to change the input to point to a different motion graphic asset. See the section called &quot;Motion Overlays via REST API&quot; (p. 180). Set the <code>&lt;active&gt;</code> tag to true and set the <code>&lt;action_time&gt;</code> tag to the time you want the start the new motion overlay. Set the <code>&lt;loop&gt;</code> tag to true or false.</td>
</tr>
<tr>
<td>Not running</td>
<td>Run a different motion overlay of a different file type.</td>
<td>Enter a Modify Overlay on Running Event command and set the input tag to point to a different type of asset, as described in the appropriate section of this chapter. Set the <code>&lt;active&gt;</code> tag to true and set the <code>&lt;action_time&gt;</code> tag to the time you want the start of the new motion overlay. Set the <code>&lt;loop&gt;</code> tag to true or false.</td>
</tr>
<tr>
<td>Running</td>
<td>Stop the running motion overlay</td>
<td>Just before you want the motion overlay to stop, enter a Modify Overlay on Running Event command to set the <code>&lt;active&gt;</code> tag to false. The motion overlay will immediately be stopped.</td>
</tr>
</tbody>
</table>
### Running

**Start the motion overlay**

To start the motion overlay again, enter a Modify Overlay on Running Event command to set the `<active>` tag to true and the `<action_time>` tag set to the time you want the overlay to begin.

**Change to a different motion overlay.**

Enter a Modify Overlay on Running Event command to change the input to a different file.

All tags, not just the ones you want to change. If you exclude a tag, the default value will apply.

---

### Runtime REST Commands Change the Event XML

When you send REST commands during an event, the event XML is permanently changed. Any data sent via the REST call goes into the XML.

For example, you might put a scoreboard overlay on your event during a sporting event. If you do not send a REST call to deactivate the overlay once the game has ended, the scoreboard will appear again at the same time the next day.

Therefore, make sure to do one of the following:

- If you plan to run the event (event A) again with different video content but with the same graphic overlays, make sure to export your XML for re-use before starting the event. Then create a new event (event B) using the exported XML. Do not start event A again.

- If you are running a 24/7 channel and you do not want your overlay to recur, remember to send a REST command to set `<activate>` to false once the overlay has run. This will delete the entire `<insertable_images>` element from the event XML.

You could also specify an absolute start time for each overlay by using the ISO 8601 UTC time format to specify an absolute time and date for the overlay. The overlays will not run again the next day.

### How to Insert a Motion Overlay with a SWF File

Use third-party software to create a SWF file. Then place this file in a directory AWS Elemental Live can access, either locally, through a mount point, or from an S3 bucket. Then set up your event to insert this file at the appropriate time on the underlying video.

In the SWF file you can optionally include Adobe ActionScript code that, at a specified time, sends a series of commands to manipulate the underlying video to provide special effects. The script can include variables and you can use REST API commands to assign values to those variables, as desired, over the runtime of the SWF file.

The motion overlay can be configured with a start time and to run only once or to loop until you optionally clear the insertion. The motion overlay can overlay the video at any position, as specified by number of pixels offset from the upper left corner.

---

### Web Interface and REST API
### Procedural Step

<table>
<thead>
<tr>
<th>Procedural Step</th>
<th>Methods Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Prepare the SWF file.</td>
<td>Use a third-party process.</td>
</tr>
<tr>
<td>2. Create the event with the desired motion overlays. As part of the creation, make sure that the live_event XML file includes a motion_image tag. Also include an action_time tag and a loop tag (set to true).</td>
<td>Use the REST API or the web interface</td>
</tr>
<tr>
<td>3. Start the event and wait for the motion overlay to start playing.</td>
<td>Use the REST API or the web interface</td>
</tr>
<tr>
<td>4. Set up more motion overlays as desired</td>
<td>Use the REST API.</td>
</tr>
<tr>
<td>5. Send updated values for swf_args as desired</td>
<td>Use the REST API.</td>
</tr>
</tbody>
</table>

The following are valid web interface and REST API combinations:

- Initial setup via web interface and run-time changes via the REST API.
- Initial setup and run-time changes via the REST API.

If you use the web interface to perform the initial setup, note the following restrictions:

- There is no mechanism for changing motion overlays at runtime via the web interface.

### Step A: Prepare the SWF Asset

1. Create a File – Use third-party software to create the SWF file. Take care with these aspects:
   - Encoder processing resources: SWF files can be very resource-intensive. See the section called "Motion Graphic Overlay with SWF" (p. 201).
   - Aspect ratio: The motion overlay can have any aspect ratio. It does not have to match the aspect ratio of the video output.
   - Size: The motion overlay can be any size, in pixels, up to the size of the underlying video. The motion overlay must be prepared in the desired size; there is no way to resize it when setting it up in the event.
   - Position: The overlay cannot be positioned so that part of the overlay runs beyond the right edge and/or bottom edge of the underlying video.
     - If you set up a motion overlay so that it is too big or it overruns and AWS Elemental Live can identify this error at event creation time, then an error message will appear.
     - If AWS Elemental Live cannot identify the error in advance, an error message will appear while the event is running. The event will not stop but the insertion request will fail.
2. Place the File – Place the SWF file in a location accessible to AWS Elemental Live: on a local directory, on a remote filesystem accessible via mount point, or in an S3 bucket.

You can specify the location in one of the following ways:

- Local to the AWS Elemental Live appliance. E.g. /data/assets/overlay.swf
- A remote server via a mount point. E.g. /data/mnt/assets/overlay.swf
- An S3 bucket, using SSL. E.g. s3ssl://company.test/sample_bucket/overlay.swf
- An S3 bucket, without SSL. E.g. s3://company.test/sample_bucket/overlay.swf

**Step B: Set up the Event**

Using the web interface, create or modify a non-running event as follows:

1. In the Global Processors section, go to the Image Inserter field and click On. More fields appear; see the table below.

2. Complete the fields as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion Mode</td>
<td>Choose SWF.</td>
</tr>
<tr>
<td>Input</td>
<td>The location and filename of the SWF file.</td>
</tr>
<tr>
<td></td>
<td>For S3, use sse=true to enable S3 Server Side Encryption (SSE) and rrs=true to enable Reduced Redundancy Storage (RRS). Default values for RRS and SSE are false.</td>
</tr>
<tr>
<td>Username Password</td>
<td>If access to your local or mounted directory requires a username and password, click the lock icon next to the Input field to show the Username and Password fields.</td>
</tr>
<tr>
<td></td>
<td>For S3, Enter the Access Key ID in the username field. Enter the Secret Access Key in the password field.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Left</td>
<td>Placement of the left edge of the motion overlay relative to the left edge of the video frame, in pixels. 0 is the left edge of the frame.</td>
</tr>
<tr>
<td></td>
<td>Take note of the width of the motion overlay and make sure that the position of the left edge of the motion overlay does not cause the right edge to be cropped.</td>
</tr>
<tr>
<td>Top</td>
<td>Placement of the top edge of the motion overlay relative to the top edge of the video frame, in pixels. 0 is the top edge of the frame. Default 0.</td>
</tr>
<tr>
<td></td>
<td>Take note of the height of the motion overlay and make sure that the position of the top edge of the motion overlay does not cause the bottom edge to be cropped.</td>
</tr>
<tr>
<td>ActionTime</td>
<td>The start time for the overlay. Specify the start time in one of the formats discussed in detail below this table.</td>
</tr>
<tr>
<td>Loop Input</td>
<td>• Check to loop the motion overlay indefinitely.</td>
</tr>
<tr>
<td></td>
<td>The motion overlay will run until the event ends. To stop the overlay earlier, see the section on modifying overlays on a running event, in the section called “Motion Overlays via REST API” (p. 180).</td>
</tr>
<tr>
<td></td>
<td>• Clear checkbox to run the motion overlay only once.</td>
</tr>
<tr>
<td>Full Frame</td>
<td>Expand the overlay to fit the video frame. In this case, make sure Left and Top are set to 0.</td>
</tr>
<tr>
<td></td>
<td>If this field is checked and the motion overlay has a different aspect ratio to the underlying video, the motion overlay will be scaled until either:</td>
</tr>
<tr>
<td></td>
<td>• The motion overlay fits in the length. The overlay will then be positioned with equal space on the left and right.</td>
</tr>
<tr>
<td></td>
<td>• The motion overlay fits in the width. The overlay will then be positioned with equal space above and below.</td>
</tr>
<tr>
<td></td>
<td>Note that the “Stretch to output” field in the Stream section does not affect the motion overlay; even if the video is stretched, the overlay is not stretched.</td>
</tr>
</tbody>
</table>
Field | Description
--- | ---
Active | Always check this box when initially setting up the motion overlay. After the initial setup, the value of this tag can be changed via the REST API to manage the content and behavior of the motion overlay.
SWF Arguments | If you included ActionScript code in the SWF asset and if that code includes arguments, enter values for the arguments in simple JSON name/value format.
Enable Rest Control | Check this field only if you plan to manage motion overlays via the REST API, after this initial setup via the web interface. Typically, you will want this tag to be true.

**Action Time Formats**

**Option 1:** Timecode format (HH:MM:SS:FF). If Timecode Source field is System Clock or Local System Clock (AWS Elemental Live only): The timecode of each frame in the input is the system time at which the frame is decoded.

The overlay start is compared to the output timecode (which is shared by all outputs). The source for the output timecode is specified for the entire event, in the Timecode Config > Source field. The output timecode is calculated as follows:

- If Source is Embedded: The timecode is extracted from the timecode carried with the input media. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.
- If Source is Start at 0: The output timecode for the first frame is 00:00:00:00 and then the output timecode counts up with each successive frame in the entire output.
- If Source is System Clock or Local System Clock (AWS Elemental Live only): The output timecode for the first frame is the system time at which the frame is decoded. Then the output timecode counts up with each successive frame in the entire output.
- If Source is Specified Start: The output timecode for the first frame is the time you specified when you selected this option as the timecode source. Then the output timecode counts up with each successive frame in the entire output.
- If Source is External Reference Connector (AWS Elemental Live only): The timecode is extracted from external LTC source. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.

**Option 2:** ISO 8601 UTC time with no dashes or colons. For example, 20160102T030405.678Z. In this case, the start time for every overlay will be the UTC time.

**Option 3:** You can only use this while adding or modifying an overlay in a running event. Set this tag to an empty string to set the start time to “now”. With this option, the start time is never exact. You cannot use this option when creating an event or modifying a non-running event.

**Step C: Manage the Overlay on a Running Event**

After the event starts, you can work with motion overlays only via the REST API. You cannot work with motion overlays on a running event via the web interface.
The motion overlay feature lets you specify only one motion overlay, but using the REST API, you can re-use it (start it again and again), change its content each time it is run, and change its attributes (for example, size) each time it runs.

1. If you did not set the enable_rest tag to true when you created the event, modify the event (PUT Live Event) and set this value to true. Do this before the event is running. See the section called “Motion Overlays via REST API” (p. 180) for the location of this tag.

2. Send the Modify Overlay on Running Event command to make the desired change to the overlays in the event. See the following table for information on the types of changes you can make to a running event via the REST API:

<table>
<thead>
<tr>
<th>Current State of Overlay</th>
<th>Desired Action</th>
<th>What to Do</th>
<th>Tags to Include in Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not running</td>
<td>Enable overlay control via the REST API</td>
<td>Enter a Modify Overlay on Running Event command to set the &lt;enable_rest&gt; tag to true, in case you did not set it to true when you created the event.</td>
<td>enable_rest; Do not include any other tags!</td>
</tr>
<tr>
<td>Not running</td>
<td>Run the motion overlay immediately</td>
<td>Enter a Modify Overlay on Running Event command to set the active tag to true and set the &lt;action_time&gt; tag to empty.</td>
<td>active action_time</td>
</tr>
<tr>
<td>Not running</td>
<td>Run the motion overlay again at a specified time</td>
<td>Enter a Modify Overlay on Running Event command to set the active tag to true and set the &lt;action_time&gt; tag to the desired start time.</td>
<td>active action_time</td>
</tr>
<tr>
<td>Not running</td>
<td>Run a different motion overlay of the same file type.</td>
<td>Enter a Modify Overlay on Running Event command to change the input to point to a different motion graphic asset. See the section called “Motion Overlays via REST API” (p. 180). Set the &lt;active&gt; tag to true and set the &lt;action_time&gt; tag to the time you want the start the new motion overlay. Set the &lt;loop&gt; tag to true or false.</td>
<td>All tags that apply to the current file type.</td>
</tr>
</tbody>
</table>
### Not running
- Run a different motion overlay of a different file type.
- Enter a Modify Overlay on Running Event command and set the input tag to point to a different type of asset, as described in the appropriate section of this chapter.
- Set the \texttt{active} tag to true and set the \texttt{action\_time} tag to the time you want the start of the new motion overlay. Set the \texttt{loop} tag to true of false.
- All tags that apply to the file type you want to run.

### Running
- Stop the running motion overlay
- Just before you want the motion overlay to stop, enter a Modify Overlay on Running Event command to set the \texttt{active} tag to false.
- The motion overlay will immediately be stopped.

### Running
- Start the motion overlay
- To start the motion overlay again, enter a Modify Overlay on Running Event command to set the \texttt{active} tag to true and the \texttt{action\_time} tag set to the time you want the overlay to begin.

### Running
- Change to a different motion overlay.
- Enter a Modify Overlay on Running Event command to change the input to a different file.
- All tags, not just the ones you want to change. If you exclude a tag, the default value will apply.

#### Runtime REST Commands Change the Event XML

When you send REST commands during an event, the event XML is permanently changed. Any data sent via the REST call goes into the XML.

For example, you might put a scoreboard overlay on your event during a sporting event. If you do not send a REST call to deactivate the overlay once the game has ended, the scoreboard will appear again at the same time the next day.

Therefore, make sure to do one of the following:
• If you plan to run the event (event A) again with different video content but with the same graphic overlays, make sure to export your XML for re-use before starting the event. Then create a new event (event B) using the exported XML. Do not start event A again.
• If you are running a 24/7 channel and you do not want your overlay to recur, remember to send a REST command to set `<activate>` to false once the overlay has run. This will delete the entire `<insertable_images>` element from the event XML.

You could also specify an absolute start time for each overlay by using the ISO 8601 UTC time format to specify an absolute time and date for the overlay. The overlays will not run again the next day.

**Graphic Overlay Plus Dynamic Content Switching**

You can combine the Graphic Overlay feature with the Dynamic Content Switching feature. You can use the Dynamic Content Switching feature to continually add and modify inputs in an event that is running. As you add inputs, you can insert static or motion overlays, as desired. For information on dynamic content switching, see Dynamic Content Switching (p. 109).

**Scheduling Inputs and Overlays**

The scheduling of the inputs and the overlays is completely decoupled.

A given input X may be added several times to the dynamic playlist, for example, so that it plays at 2:00 p.m. and then plays again at 3:10 p.m.

If you want an overlay to appear the first time input X plays, you might set its start time for 2:10 p.m. The next time that input X plays, there is no logic to play the same overlay again because 2:10 p.m. has passed. Therefore, if you want the overlay to appear again on input X, you must send a Modify Overlay call again.

**Behavior with Different Insertion Options for Static Overlays**

Points to remember when using Static Graphic Overlay with Dynamic Content Switching:

1. **Playlist input plus overlay at the input stage.**

   The overlay is specified in the Input section of the event. Therefore, however many times you include this input in the playlist, the overlay will be included unless you action to remove it.
   - If you want to include the overlay in a given repetition of the input, you may need to change the start time of the overlay. See the information on the Start Time in the table under the section called “Step B” (p. 144).
   - If you do not want to include the overlay in a given repetition of that input, you must use the REST interface to enter a Modify Static Overlay command and delete the overlay. See the section called “Commands” (p. 171).

2. **Playlist input plus overlay at the global processing stage.**

   The scheduling of the inputs and the overlays is completely decoupled.

   A given input X may be added several times to the dynamic playlist, for example, so that it plays at 2:00 p.m. and then plays again at 3:10 p.m.

   If you want an overlay to appear the first time input X plays, you might set its start time for 2:10 p.m. The next time that input X plays, there is no logic to play the same overlay again because 2:10 p.m. has passed.

   Therefore, if you want the overlay to appear again on input X, you must enter the Modify Static Overlay command again.
3. Playlist input plus overlay at the output stage. The same comments as for global processing stage apply.

Using the REST API for Static Overlays

Topics
- Static Graphic Overlay Commands (p. 171)
- Create or Modify a Non-Running Event with Static Graphic Overlay (p. 172)
- Modify Static Overlay on a Running Event (p. 179)

Static Graphic Overlay Commands

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Action</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Event</td>
<td>POST</td>
<td>&lt;Live IP address&gt;/live_events</td>
<td>Create an event that includes static overlay information.</td>
</tr>
<tr>
<td>Modify Event</td>
<td>PUT</td>
<td>&lt;Live IP address&gt;/live_events/live_event/&lt;event ID&gt;</td>
<td>Modify an event (that is not running) and add, modify or delete static overlay information.</td>
</tr>
<tr>
<td>Modify Static Overlay, Running Event</td>
<td>POST</td>
<td>/live_event/&lt;event ID&gt;/image_inserter</td>
<td>All outputs: add, modify or delete static information in a running event.</td>
</tr>
<tr>
<td>Modify Static Overlay, Running Event</td>
<td>POST</td>
<td>&lt;Live IP Address&gt;/live_events/&lt;event ID&gt;/image_inserter/output/&lt;output id&gt;</td>
<td>Specific output: add, modify or delete static information in a running event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Where &lt;output id&gt; is the unique ID automatically assigned to this output when the event is created.</td>
<td></td>
</tr>
<tr>
<td>Modify Static Overlay, Running Event</td>
<td>POST</td>
<td>&lt;Live IP Address&gt;/live_events/&lt;event ID&gt;/image_inserter/output/by_stream/&lt;stream id&gt;</td>
<td>All outputs associated with a specific stream: add, modify or delete static information in a running event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Where &lt;stream id&gt; is the unique ID automatically assigned to this stream when the event is created. The ID can change while the event is running.</td>
<td></td>
</tr>
</tbody>
</table>
Create or Modify a Non-Running Event with Static Graphic Overlay

Create or modify an AWS Elemental Live event and include one or more static graphic overlays. This description assumes that you are familiar with the XML body for a live_event aside from the data for the graphic overlay.

HTTP Request and Response

HTTP URL

POST http://<Live IP Address>/live_events

or:

PUT http://<Live IP Address>/live_events/<event ID>

Body of Request – One Input
To insert the static overlay in one input element:

XML content consisting of one live_event element that contains:

- All the usual elements and tags.
- One input element that contains:
  - All the usual elements and tags.
  - One image_inserter element that contains:
    - One enable_rest tag.
    - One insertable_images element that contains:
      - 1 to 8 insertable_image elements that contain the tags listed in the following table.

For a representation of the XML structure, see the section called “XML Structure” (p. 196).

Body of Request – All Outputs

To insert the static overlay in all outputs:

XML content consisting of one live_event element that contains:

- All the usual elements and tags.
- One image_inserter element that contains:
  - One enable_rest tag.
  - One insertable_images element that contains:
    - 1 to 8 insertable_image elements that contain the tags listed in the table on the following page.

For a representation of the XML structure, see the section called “XML Structure” (p. 196).

Body of Request – Outputs for One Stream Assembly

To insert the static overlay in the outputs associated with a given stream. XML content consisting of one live_event element that contains:

- All the usual elements and tags.
- One stream_assembly element that contains:
  - All the usual elements and tags.
  - One image_inserter element that contains:
    - One enable_rest tag.
    - One insertable_images element that contains:
      - 1 to 8 insertable_image elements that contain the tags listed in the following table.

For a representation of the XML structure, see the section called “XML Structure” (p. 196).

Child Elements of <insertable_image> and <image>

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Required</th>
<th>Description for Creating</th>
</tr>
</thead>
<tbody>
<tr>
<td>activate</td>
<td>Boolean</td>
<td>Required</td>
<td>Required when adding or modifying an overlay in a running event (not when creating an event)</td>
</tr>
</tbody>
</table>

Version 2.17
173
<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Required</th>
<th>Description for Creating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>running event). This tag has no effect when creating or modifying a non-running event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Set to true when adding any overlay or modifying an overlay (in specified the layer) in an existing event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Set to false to delete the overlay (specified in the layer tag) from the underlying video. Note that the entire overlay is also deleted from the event XML. (If you do not specify a layer tag in the body, the overlay won’t be deleted.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> <code>&lt;activate&gt;</code> is distinct from <code>&lt;active&gt;</code>, which is used with motion image inserter.</td>
</tr>
<tr>
<td>duration</td>
<td>Integer</td>
<td>Optional Default: until end of event</td>
<td>The time in milliseconds for the overlay to remain on the video.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When creating an event, if this field is left blank, the overlay will remain on the video as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• In the Input section: Until this input ends.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• In the Global Processor section: Until the event ends.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• In the Output section: Until the event ends.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The total running time of the overlay is <code>fade_in + duration + fade_out</code>.</td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Required</td>
<td>Description for Creating</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>----------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>fade_in</td>
<td>Integer</td>
<td>Optional</td>
<td>The duration, in milliseconds, for the overlay fade-in. This time is inserted before the start time of the overlay.</td>
</tr>
<tr>
<td>fade_out</td>
<td>Integer</td>
<td>Optional</td>
<td>This field is valid only if the Duration field is completed. The duration, in milliseconds, for the overlay fade-out. This time is added to the overlay duration.</td>
</tr>
<tr>
<td>height</td>
<td>Integer</td>
<td>Optional Default: native height of overlay</td>
<td>The height of the overlay when inserted in the video, in pixels. When creating an event, leave blank to use the native height of the overlay. The original overlay will be scaled up or down, to the specified height.</td>
</tr>
<tr>
<td>width</td>
<td>Integer</td>
<td>Optional Default: native width of overlay</td>
<td>The width of the overlay when inserted in the video, in pixels. When creating an event, leave blank to use the native height of the overlay. The original overlay will be scaled up or down, to the specified width.</td>
</tr>
<tr>
<td>image_inserter_input</td>
<td>Location</td>
<td>Required</td>
<td>Overlay PNG or BMP file to insert. For file requirement details, information about where to store this file, and how to specify its location, see the section called “Step A” (p. 144).</td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Required</td>
<td>Description for Creating</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>----------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>image_x</td>
<td>Integer</td>
<td>Required</td>
<td>Placement of the left edge of the overlay relative to the horizontal axis for the video frame, in pixels. 0 is the left edge of the frame. When creating an event, cannot be omitted. Take note of the width of the overlay and make sure that the position of the left edge of the overlay does not cause the right edge to be cropped.</td>
</tr>
<tr>
<td>image_y</td>
<td>Integer</td>
<td>Required</td>
<td>Placement of the top edge of the overlay relative to the vertical axis for the video frame, in pixels. 0 is the top edge of the frame. When creating an event, cannot be omitted. Take note of the height of the overlay and make sure that the position of the top edge of the overlay does not cause the bottom edge to be cropped.</td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Required</td>
<td>Description for Creating</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td>----------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>layer</td>
<td>Integer</td>
<td>Required</td>
<td>A number from 0 to 7 to specify the Z order of the inserted overlay. “Z order” means that overlays with higher values of layer will be inserted on top of overlays with lower values of layer. Must always be specified. In the XML for modifying a static overlay at runtime, if the XML has more than one image container, then each layer tag must have a different value. So each overlay must be in its own layer.</td>
</tr>
<tr>
<td>opacity</td>
<td>Integer</td>
<td>Optional</td>
<td>The opacity of the overlay, as a number from 0 to 100. 0 is transparent. 100 is fully opaque. When creating an event, cannot be omitted.</td>
</tr>
<tr>
<td>start_time</td>
<td>String</td>
<td>Optional</td>
<td>The start time for the overlay. Specify the start time in one of the formats discussed below this table.</td>
</tr>
</tbody>
</table>

**Start Time Formats**

**Option 1**: Timecode format (HH:MM:SS:FF). The overlay start is determined by comparing the specified start to the appropriate timecode.

- If the overlay is specified in the Input section: The start time for the static overlay will be compared to the input timecode (the timecode for a given input). The source for the input timecode is specified separately for each input (Input > Timecode Source field). The input timecode is calculated as follows:
  - If Timecode Source is Embedded: The timecode associated with each frame is extracted from the timecode carried with the input media. Note that each input will have its own timecode and the timecode may not align well from one input to another.
  - If Timecode Source field is Start at 0: The timecode of the first frame of the input is 00:00:00:00 and the timecode counts up with each successive frame. The timecode starts over with each input.
  - If Timecode Source field is System Clock or Local System Clock (AWS Elemental Live only): The timecode of each frame in the input is the system time at which the frame is decoded.
If the overlay is specified in the Global Processor section: The overlay start is compared to the output timecode (which is shared by all outputs). The source for the output timecode is specified for the entire event, in the Timecode Config > Source field. The output timecode is calculated as follows:

- **If Source is Embedded:** The timecode is extracted from the timecode carried with the input media. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.
- **If Source is Start at 0:** The output timecode for the first frame is 00:00:00:00 and then the output timecode counts up with each successive frame in the entire output.
- **If Source is System Clock or Local System Clock (AWS Elemental Live only):** The output timecode for the first frame is the system time at which the frame is decoded. Then the output timecode counts up with each successive frame in the entire output.
- **If Source is Specified Start:** The output timecode for the first frame is the time you specified when you selected this option as the timecode source. Then the output timecode counts up with each successive frame in the entire output.
- **If Source is External Reference Connector (AWS Elemental Live only):** The timecode is extracted from external LTC source. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.

If the static overlay is specified in the Output section: The start time for the static overlay is calculated in the same way as a static overlay in the Global Processor section.

### Option 2
ISO 8601 UTC time with no dashes or colons. For example, 20160102T030405.678Z. In this case, the start time for every overlay (regardless of whether it is defined in the input, the global processor or the output) will be the UTC time.

### Option 3
Only when adding or modifying an overlay in a running event (not when creating an event or modifying a non-running event), set this tag to an empty string to set the start time to “now”. With this option, the start time is never exact.

### Example

The following request creates an event with the name myLiveEvent and includes one static overlay to insert the file logo.bmp. The overlay is inserted directly inside the live_event, which means it will appear in all outputs.

```
POST http://198.51.100.22/live_events

<?xml version="1.0" encoding="UTF-8"?>
<live_event>
  <name>myLiveEvent</name>
  ...
  <image_inserter>
    <enable_rest>true</enable_rest>
    <insertable_images>
      <insertable_image>
        <duration>30000</duration>
        <fade_in>10</fade_in>
        <fade_out>10</fade_out>
        <height>900</height>
        <left>300</left>
        <top>400</top>
        <layer>0</layer>
        <start_time>16:09:09:10</start_time>
        <width>800</width>
        <image_inserter_input>
          <uri>mnt/storage/logo.bmp</uri>
        </image_inserter_input>
      </insertable_image>
    </insertable_images>
  </image_inserter>
</live_event>
```
Modify Static Overlay on a Running Event

In a running event, you can use the REST API to add more static overlays, modify the behavior of an existing static overlay, or delete an existing static overlay.

**Note**

Commands sent during an event change the event XML. Therefore, if you export the XML and create a new event with it, the new event will have any overlays set up as they were set during the course of the event, not as they were when the event was created.

HTTP Request and Response

**HTTP URL - One Input**

To add, modify, or delete the static overlay in one input element.

```
POST http://<Live IP Address>/live_events/<id>/image_inserter/input/<input id>
```

Where `<input id>` is the unique ID automatically assigned to this input when the event is created or when the input is added to the event.

or:

```
POST http://<Live IP Address>/live_events/<id>/image_inserter/input/by_label/<input label>
```

Where `<input label>` is the input label you assigned when you created this event or created this input. Input labels are always optional.

**HTTP URL - All Outputs**

To add, modify, or delete the static overlay in all outputs.

```
POST http://<Live IP Address>/live_events/<id>/image_inserter
```

**HTTP URL - One Output**

To add, modify, or delete the static overlay in one output.

```
POST http://<Live IP Address>/live_events/<id>/image_inserter/output/<output id>
```

Where `<output id>` is the unique ID automatically assigned to this output when the event is created.

**HTTP URL - Outputs for One Stream Assembly**

To add, modify, or delete the static overlay in the outputs associated with a given stream.

```
POST http://<Live IP Address>/live_events/<id>/image_inserter/output/by_stream/<stream id>
```
Where `<stream_id>` is the `<ID>` automatically assigned to this stream when the event is created. The ID can change while the event is running (for example, if another stream is deleted), so you may need to obtain the current ID before sending this command.

**Body of Request**

XML content consisting of:

- One or more `image_inserter` elements that each contains:
  - 1 to 8 `<image>` elements that each contains the tags in the table in the section [“Create or Modify a Non-Running Event”](p. 172).

For information on the tags to include for different actions, see the section called [“Types of Changes”](p. 149).

**Response**

The response repeats back the data that you posted with `<ID>` tags for `image_inserter` and each overlay. If the event is not running, the message “Event `<ID>` is not running” is returned.

**Example**

The following request modifies the overlays in the event with the ID 16. It modifies the start time on the existing overlay in layer 3.

It also adds one overlay (in layer 4); if an overlay already exists in layer 4, that overlay is replaced with the new overlay (even if that overlay is currently running). If an overlay does not exist in layer 4, the overlay is added.

```
POST http://198.51.100.22/live_events/16/image_inserter

<?xml version="1.0" encoding="UTF-8"?>
<image_inserter>
  <image>
    <activate>true</activate>
    <layer>3</layer>
    <start_time>17:09:09:10</start_time>
  </image>
  <image>
    <activate>true</activate>
    <duration>30000</duration>
    <fade_in>10</fade_in>
    <fade_out>10</fade_out>
    <height>900</height>
    <left>300</left>
    <top>400</top>
    <layer>4</layer>
    <start_time>16:09:09:10</start_time>
    <width>800</width>
    <image_inserter_input>
      <uri>mnt/storage/logo.bmp</uri>
    </image_inserter_input>
  </image>
</image_inserter>
```

**Using the REST API for Motion Overlays**

**Topics**

Version 2.17

180
Motion Overlay Commands

Motion Overlay with MOV

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Action</th>
<th>Signature</th>
<th>Description</th>
<th>See...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Event</td>
<td>POST</td>
<td><code>&lt;Live IP address&gt;/live_events</code></td>
<td>Create an event that includes MOV motion overlay information.</td>
<td>the section called “Motion Overlay with MOV” (p. 182)</td>
</tr>
<tr>
<td>Modify Non-Running Event</td>
<td>PUT</td>
<td><code>&lt;Live IP address&gt;/live_events/&lt;ID of live_event&gt;</code></td>
<td>Modify an event (that is not running) and add, modify or delete MOV motion overlay information.</td>
<td>the section called “Motion Overlay with MOV” (p. 182)</td>
</tr>
<tr>
<td>Modify Overlay on Running Event</td>
<td>POST</td>
<td><code>&lt;Live IP address&gt;/live_events/&lt;ID of live_event&gt;/motion_image_inserter</code></td>
<td>Add, modify or delete MOV motion overlay information in a running event.</td>
<td>the section called “Modify Running Event” (p. 186)</td>
</tr>
</tbody>
</table>

Motion Overlay with PNG

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Action</th>
<th>Signature</th>
<th>Description</th>
<th>See...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Event</td>
<td>POST</td>
<td><code>&lt;Live IP address&gt;/live_events</code></td>
<td>Create an event that includes PNG motion overlay information.</td>
<td>the section called “Motion Overlay with PNG” (p. 186)</td>
</tr>
<tr>
<td>Modify Non-Running Event</td>
<td>PUT</td>
<td><code>&lt;Live IP address&gt;/live_events/&lt;ID of live_event&gt;</code></td>
<td>Modify an event (that is not running) and add, modify or delete PNG motion overlay information.</td>
<td>the section called “Motion Overlay with PNG” (p. 186)</td>
</tr>
<tr>
<td>Modify Overlay on Running Event</td>
<td>POST</td>
<td><code>&lt;Live IP address&gt;/live_events/&lt;ID of live_event&gt;/motion_image_inserter</code></td>
<td>Add, modify or delete motion information in a running event.</td>
<td>the section called “Modify on a Running Event” (p. 190)</td>
</tr>
</tbody>
</table>

Motion Overlay with SWF

Version 2.17
181
### Working with Motion Overlays with MOV Asset

#### Create or Modify a Non-Running Event with MOV Motion Overlay

When you create an AWS Elemental Live event, you can include one or more static overlays and one motion overlay. To include the motion overlay, use the REST commands in this section. This description assumes that you are familiar with the XML body for a `live_event` aside from the data for the motion graphic overlay.

**HTTP Request and Response**

**HTTP URL**

- POST `http://<Live IP Address>/live_events`  

or:

- PUT `http://<Live IP Address>/live_events/<live_event_id>`

**Body of Request**

XML content consisting of one `live_event` element that contains:

- All the usual elements and tags.
- One `motion_image_inserter` element directly inside the `live_event`. This element contains the tags listed in the table in the section [Create or Modify a Non-Running Event](#) (p. 182).

**Response**

The response repeats back the data that you posted with `<ID>` tags for many elements including IDs for the following `motion_image_inserter` elements:

- `motion_image_inserter`
- **motion_image_inserter_input**

**Child Elements of `<motion_image_inserter>`**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;action_time&gt;</code></td>
<td>String</td>
<td>The start time for the motion overlay. Specify the start time in one of the formats described below this table.</td>
</tr>
<tr>
<td><code>&lt;active&gt;</code></td>
<td>Boolean</td>
<td>Always set to true when initially setting up the motion overlay. After the initial setup, the value of this tag can be changed (possibly in combination with action_time) to start and stop the motion overlay. Note: <code>&lt;active&gt;</code> is distinct from <code>&lt;activate&gt;</code>, which is used for static image overlays.</td>
</tr>
<tr>
<td><code>&lt;enable_rest&gt;</code></td>
<td>Boolean</td>
<td>When creating or modifying a non-running event, set to true if you plan to manage motion overlays via the REST API, after the initial setup. When modifying the motion overlay in a running event, you typically want to leave this tag set to true. Set to false only if you want to block the ability to work with motion overlay in a running event.</td>
</tr>
</tbody>
</table>
| `<full_frame>` | Boolean       | Expand the overlay to fit the video frame. In this case, make sure Left and Top are set to 0 or left blank. If this field is checked and the motion overlay has a different aspect ratio to the underlying video, the motion overlay will be scaled until either:  
  - The motion overlay fits in the length. The overlay will then be positioned with equal space on the left and right.  
  - The motion overlay fits in the width. The overlay will then be positioned with equal space above and below. |
<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Note that the “Stretch to output” field in the Stream section does not affect the motion overlay; even if the video is stretched, the overlay is not stretched.</td>
</tr>
<tr>
<td>&lt;image_x&gt;</td>
<td>Integer</td>
<td>Placement of the left edge of the motion overlay relative to the left edge of the video frame, in pixels. 0 is the left edge of the frame. When creating an event, cannot be omitted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Take note of the width of the motion overlay and make sure that the position of the left edge of the motion overlay does not cause the right edge to be cropped.</td>
</tr>
<tr>
<td>&lt;image_y&gt;</td>
<td>Integer</td>
<td>Placement of the top edge of the motion overlay relative to the top edge of the video frame, in pixels. 0 is the top edge of the frame. When creating an event, cannot be omitted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Take note of the height of the motion overlay and make sure that the position of the top edge of the motion overlay does not cause the bottom edge to be cropped.</td>
</tr>
<tr>
<td>&lt;insertion_mode&gt;</td>
<td>String</td>
<td>Enter mov.</td>
</tr>
<tr>
<td>&lt;loop_input&gt;</td>
<td>Boolean</td>
<td>• True to loop the motion overlay indefinitely. The motion overlay will run until the event ends. To stop the overlay earlier, send the Modify MOV Motion Overlay command with &lt;active&gt; set to false.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• False to run the motion overlay once.</td>
</tr>
<tr>
<td>&lt;motion_image_inserter_input&gt;</td>
<td>Child element</td>
<td>&lt;motion_image_inserter_input&gt; contains only the &lt;uri&gt; element, described in the next row.</td>
</tr>
<tr>
<td>&lt;uri&gt;</td>
<td>String</td>
<td>The path and filename of the MOV asset.</td>
</tr>
</tbody>
</table>

**Action Time Formats**
**Option 1:** Timecode format (HH:MM:SS:FF). If Timecode Source field is System Clock or Local System Clock (AWS Elemental Live only): The timecode of each frame in the input is the system time at which the frame is decoded.

The overlay start is compared to the output timecode (which is shared by all outputs). The source for the output timecode is specified for the entire event, in the Timecode Config > Source field. The output timecode is calculated as follows:

- If Source is Embedded: The timecode is extracted from the timecode carried with the input media. That timecode becomes the output timecode for the first transoded frame. Then the output timecode counts up with each successive frame in the entire output.
- If Source is Start at 0: The output timecode for the first frame is 00:00:00:00 and then the output timecode counts up with each successive frame in the entire output.
- If Source is System Clock or Local System Clock (AWS Elemental Live only): The output timecode for the first frame is the system time at which the frame is decoded. Then the output timecode counts up with each successive frame in the entire output.
- If Source is Specified Start: The output timecode for the first frame is the time you specified when you selected this option as the timecode source. Then the output timecode counts up with each successive frame in the entire output.
- If Source is External Reference Connector (AWS Elemental Live only): The timecode is extracted from external LTC source. That timecode becomes the output timecode for the first transoded frame. Then the output timecode counts up with each successive frame in the entire output.

**Option 2:** ISO 8601 UTC time with no dashes or colons. For example, 20160102T030405.678Z. In this case, the start time for every overlay will be the UTC time.

**Option 3:** You can only use this while adding or modifying an overlay in a running event. Set this tag to an empty string to set the start time to “now”. With this option, the start time is never exact. You cannot use this option when creating an event or modifying a non-running event.

**Example**

The following request creates an event with the name myLiveEvent and includes a motion_image_inserter section that inserts the motion overlay represented by a set of logo.png files:

```xml
POST http://198.51.100.22/live_events

<?xml version="1.0" encoding="UTF-8"?>
<live_event>
  <name>myLiveEvent</name>
  <motion_image_inserter>
    <action_time>10:16:23:10</action_time>
    <active>true</active>
    <enable_rest>true</enable_rest>
    <framerate_denominator>1001</framerate_denominator>
    <framerate_numerator>30000</framerate_numerator>
    <full_frame>false</full_frame>
    <left>100</left>
    <top>200</top>
    <insertion_mode>mov</insertion_mode>
    <loop_input>true</loop_input>
    <motion_image_inserter_input>
      <uri>/data/logo001.mov</uri>
    </motion_image_inserter_input>
  </motion_image_inserter>
</live_event>
```
Modify MOV Motion Overlay on a Running Event

In a running event, modify the content and/or behavior of the existing MOV motion overlay.

**Note**

Commands sent during an event change the event XML. Therefore, if you export the XML and create a new event with it, the new event will have any overlays set up as they were set during the course of the event, not as they were when the event was created.

**HTTP Request and Response**

**HTTP URL**

```plaintext
POST http://<Live IP Address>/live_events/<live event id>/motion_image_inserter
```

**Body of Request**

XML content consisting of one motion_image_inserter element that contains the required tags. For the tags, see the table in the section the section called “Create or Modify a Non-Running Event” (p. 182). For information on the tags to include for different actions, see the section called “Motion Graphic Overlay” (p. 150).

**Response**

The response repeats back the data that you posted with <ID> tags for motion_image_inserter and motion_image_inserter_input.

**Example**

The following example request modifies the running event with the ID 33. It sets up the currently defined overlay MOV to run again at 20160102T030405.678Z.

```plaintext
POST http://198.51.100.22/live_events/33/motion_image_inserter

-------------------------------------------------------------
<motion_image_inserter>
   <action_time>20160102T030405.678Z</action_time>
   <active>true</active>
</motion_image_inserter>
```

Working with Motion Overlays with PNG Asset

Create or Modify a Non-Running Event with PNG Motion Overlay

Create an AWS Elemental Live event and include one or more static overlays or one motion overlay. This description assumes that you are familiar with the XML body for a live_event aside from the data for the motion graphic overlay.

**HTTP Request and Response**

**HTTP URL**

```plaintext
POST http://<Live IP Address>/live_events
```

or

```plaintext
PUT http://<Live IP Address>/live_events/<live event id>
```

**Body of Request**

XML content consisting of one live_event element that contains:
• All the usual elements and tags.
• One motion_image_inserter element directly inside the live_event. This element contains the tags listed in the following table.

Response

The response repeats back the data that you posted with <ID> tags for many elements including IDs for the following motion image inserter elements:

• motion_image_inserter
• motion_image_inserter_input

Child Elements of <motion_image_inserter>

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;action_time&gt;</td>
<td>String</td>
<td>The start time for the motion overlay. Specify the start time in one of the formats described below this table.</td>
</tr>
<tr>
<td>&lt;active&gt;</td>
<td>Boolean</td>
<td>Always set to true when initially setting up the motion overlay. After the initial setup, the value of this tag can be changed (possibly in combination with action_time) to start and stop the motion overlay. <strong>Note:</strong> &lt;active&gt; is distinct from &lt;activate&gt;, which is used for static image overlays.</td>
</tr>
<tr>
<td>&lt;enable_rest&gt;</td>
<td>Boolean</td>
<td>When creating or modifying a non-running event, set to true if you plan to manage motion overlays via the REST API, after the initial setup. When modifying the motion overlay in a running event, you typically want to leave this tag set to true. Set to false only if you want to block the ability to work with motion overlay in a running event.</td>
</tr>
<tr>
<td>&lt;full_frame&gt;</td>
<td>Boolean</td>
<td>Expand the overlay to fit the video frame. In this case, make sure &lt;image_x&gt; and &lt;image_y&gt; are set to 0 or left blank. If this field is checked and the motion overlay has a different aspect ratio to the underlying</td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Video, the motion overlay will be scaled until either:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The motion overlay fits in the length. The overlay will then be positioned with equal space on the left and right.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The motion overlay fits in the width. The overlay will then be positioned with equal space above and below.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note that the “Stretch to output” field in the Stream section does not affect the motion overlay; even if the video is stretched, the overlay is not stretched.</td>
</tr>
<tr>
<td>&lt;image_x&gt;</td>
<td>Integer</td>
<td>Placement of the left edge of the motion overlay relative to the left edge of the video frame, in pixels. 0 is the left edge of the frame. When creating an event, cannot be omitted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Take note of the width of the motion overlay and make sure that the position of the left edge of the motion overlay does not cause the right edge to be cropped.</td>
</tr>
<tr>
<td>&lt;image_y&gt;</td>
<td>Integer</td>
<td>Placement of the top edge of the motion overlay relative to the top edge of the video frame, in pixels. 0 is the top edge of the frame. When creating an event, cannot be omitted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Take note of the height of the motion overlay and make sure that the position of the top edge of the motion overlay does not cause the bottom edge to be cropped.</td>
</tr>
<tr>
<td>&lt;insertion_mode&gt;</td>
<td>String</td>
<td>Enter png.</td>
</tr>
</tbody>
</table>

Version 2.17
Element | Type | Description
--- | --- | ---
<loop_input> | Boolean | • True to loop the motion overlay indefinitely. The motion overlay will run until the event ends. To stop the overlay earlier, send the Modify PNG Motion Overlay command with <active> set to false.
• False to run the motion overlay once.
<motion_image_inserter_input> | Child element | <motion_image_inserter_input> contains only the <uri> element, described in the next row.
<uri> | String | Provide the path and filename of the first PNG file in the series. All files in the series must have the same number of digits in the numerical part of the filename.

For example, if the files are stored on /mnt/storage/motion_logos/ and the files are named logo_hi_001 to logo_hi_357, enter “/mnt/storage/motion_logos/logo_hi_001”.

**Action Time Formats**

**Option 1:** Timecode format (HH:MM:SS:FF). If Timecode Source field is System Clock or Local System Clock (AWS Elemental Live only): The timecode of each frame in the input is the system time at which the frame is decoded.

The overlay start is compared to the output timecode (which is shared by all outputs). The source for the output timecode is specified for the entire event, in the Timecode Config > Source field. The output timecode is calculated as follows:

- If Source is Embedded: The timecode is extracted from the timecode carried with the input media. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.
- If Source is Start at 0: The output timecode for the first frame is 00:00:00:00 and then the output timecode counts up with each successive frame in the entire output.
- If Source is System Clock or Local System Clock (AWS Elemental Live only): The output timecode for the first frame is the system time at which the frame is decoded. Then the output timecode counts up with each successive frame in the entire output.
- If Source is Specified Start: The output timecode for the first frame is the time you specified when you selected this option as the timecode source. Then the output timecode counts up with each successive frame in the entire output.
- If Source is External Reference Connector (AWS Elemental Live only): The timecode is extracted from external LTC source. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.
Option 2: ISO 8601 UTC time with no dashes or colons. For example, 20160102T030405.678Z. In this case, the start time for every overlay will be the UTC time.

Option 3: You can only use this while adding or modifying an overlay in a running event. Set this tag to an empty string to set the start time to “now”. With this option, the start time is never exact. You cannot use this option when creating an event or modifying a non-running event.

Example

The following request creates an event with the name myLiveEvent and includes a motion_image_inserter section that inserts the motion overlay represented by a set of logo.png files:

```
POST http://198.51.100.22/live_events
-----------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<live_event>
  <name>myLiveEvent</name>
  ...
  <motion_image_inserter>
    <action_time>10:16:23:10</action_time>
    <active>true</active>
    <enable_rest>true</enable_rest>
    <framerate_denominator>1001</framerate_denominator>
    <framerate_numerator>30000</framerate_numerator>
    <full_frame>false</full_frame>
    <left>100</left>
    <top>200</top>
    <insertion_mode>png</insertion_mode>
    <loop_input>true</loop_input>
    <motion_image_inserter_input>
      <uri>/data/logo001.png</uri>
    </motion_image_inserter_input>
  </motion_image_inserter>
</live_event>
```

Modify PNG Motion Overlay on a Running Event

In a running event, modify the content and/or behavior of the existing PNG motion overlay.

Note

Commands sent during an event change the event XML. Therefore, if you export the XML and create a new event with it, the new event will have any overlays set up as they were set during the course of the event, not as they were when the event was created.

HTTP Request and Response

HTTP URL

```
POST http://<Live IP Address>/live_events/<live event id>/motion_image_inserter
```

Body of Request

XML content consisting of one motion_image_inserter element that contains the required tags. For the tags, see the table in the section called “Create or Modify a Non-Running Event” (p. 182). For information on the tags to include for different actions, see the section called “Motion Graphic Overlay” (p. 150). This information is in a table at the end of each section.

Response

The response repeats back the data that you posted with <ID> tags for motion_image_inserter and motion_image_inserter_input.
Example

The following example request modifies the running event with the ID 33. It sets up the currently defined overlay PNG to run again at 20160102T030405.678Z.

```xml
POST http://198.51.100.22/live_events/33/motion_image_inserter

<motion_image_inserter>
  <action_time>20160102T030405.678Z</action_time>
  <active>true</active>
</motion_image_inserter>
```

Working with Motion Overlays with SWF Asset

Create or Modify a Non-Running Event with SWF Motion Overlay

Create an AWS Elemental Live event and include one or more static overlays or one motion overlay. This description assumes that you are familiar with the XML body for a live_event aside from the data for image insertion.

HTTP Request and Response

HTTP URL

POST http://<Live IP Address>/live_events

or

PUT http://<Live IP Address>/live_events/<live event id>

Body of Request

XML content consisting of one live_event element that contains:

- All the usual elements and tags.
- One motion_image_inserter element directly inside the live_event. This element contains the tags listed in the following table.

Response

The response repeats back the data that you posted with <ID> tags for many elements including IDs for the following motion image inserter elements:

- motion_image_inserter
- motion_image_inserter_input

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;action_time&gt;</td>
<td>String</td>
<td>The start time for the motion overlay. Specify the start time in one of the formats described below this table.</td>
</tr>
<tr>
<td>&lt;active&gt;</td>
<td>Boolean</td>
<td>Always set to true when initially setting up the motion overlay.</td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>After the initial setup, the value of this tag can be changed (possibly in combination with action_time) to start and stop the motion overlay.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> <code>&lt;active&gt;</code> is distinct from <code>&lt;activate&gt;</code>, which is used for static image overlays.</td>
</tr>
<tr>
<td><code>&lt;duration&gt;</code></td>
<td>Integer</td>
<td>The time, in milliseconds, for the motion overlay to remain on the video. If left blank, the overlay plays for the full length of the .swf file.</td>
</tr>
<tr>
<td><code>&lt;enable_rest&gt;</code></td>
<td>Boolean</td>
<td>When creating or modifying a non-running event, set to true if you plan to manage motion overlays via the REST API, after the initial setup.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When modifying the motion overlay in a running event, you typically want to leave this tag set to true. Set to false only if you want to block the ability to work with motion overlay in a running event.</td>
</tr>
</tbody>
</table>
### Element Type Description

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;full_frame&gt;</code></td>
<td>Boolean</td>
<td>Expand the overlay to fit the video frame. In this case, make sure <code>&lt;image_x&gt;</code> and <code>&lt;image_y&gt;</code> are set to 0 or left blank.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If this field is checked and the motion overlay has a different aspect ratio to the underlying video, the motion overlay will be scaled until either:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The motion overlay fits in the length. The overlay will then be positioned with equal space on the left and right.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The motion overlay fits in the width. The overlay will then be positioned with equal space above and below.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note that the “Stretch to output” field in the Stream section does not affect the motion overlay; even if the video is stretched, the overlay is not stretched.</td>
</tr>
<tr>
<td><code>&lt;image_x&gt;</code></td>
<td>Integer</td>
<td>Placement of the left edge of the motion overlay relative to the left edge of the video frame, in pixels. 0 is the left edge of the frame. When creating an event, cannot be omitted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Take note of the width of the motion overlay and make sure that the position of the left edge of the motion overlay does not cause the right edge to be cropped.</td>
</tr>
<tr>
<td><code>&lt;image_y&gt;</code></td>
<td>Integer</td>
<td>Placement of the top edge of the motion overlay relative to the top edge of the video frame, in pixels. 0 is the top edge of the frame. When creating an event, cannot be omitted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Take note of the height of the motion overlay and make sure that the position of the top edge of the motion overlay does not cause the bottom edge to be cropped.</td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&lt;insertion_mode&gt;</td>
<td>String</td>
<td>Enter swf.</td>
</tr>
</tbody>
</table>
| <loop_input>                | Boolean     | • True to loop the motion overlay indefinitely. The motion overlay will run until the event ends. To stop the overlay earlier, send the Modify SWF Motion Overlay command with <active> set to false.  
  • False to run the motion overlay once. |
| <motion_image_inserter_input> | Child element | <motion_image_inserter_input> contains only the <uri> element, described in the next row.                                                   |
| <uri>                       | String      | The path and filename of the SWF asset.                                                                                                     |
| <swf_arguments>             | String      | If you included ActionScript code in the SWF asset and if that code includes arguments, enter values for the arguments in simple JSON name/value format. Each time new <swf_arguments> are transmitted (each time a Modify SWF Motion Overlay command is performed to modify swf_arguments), the flashvars of the SWF will be changed. When using ActionScript 3, all old values will be removed and these new arguments will be added, and an event (event type Event.CHANGE) will be dispatched from the root SWF's LoaderInfo object. |

**Action Time Formats**

**Option 1:** Timecode format (HH:MM:SS:FF). If Timecode Source field is System Clock or Local System Clock (AWS Elemental Live only): The timecode of each frame in the input is the system time at which the frame is decoded.

The overlay start is compared to the output timecode (which is shared by all outputs). The source for the output timecode is specified for the entire event, in the Timecode Config > Source field. The output timecode is calculated as follows:

- If Source is Embedded: The timecode is extracted from the timecode carried with the input media. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.
- If Source is Start at 0: The output timecode for the first frame is 00:00:00:00 and then the output timecode counts up with each successive frame in the entire output.
• If Source is System Clock or Local System Clock (AWS Elemental Live only): The output timecode for the first frame is the system time at which the frame is decoded. Then the output timecode counts up with each successive frame in the entire output.
• If Source is Specified Start: The output timecode for the first frame is the time you specified when you selected this option as the timecode source. Then the output timecode counts up with each successive frame in the entire output.
• If Source is External Reference Connector (AWS Elemental Live only): The timecode is extracted from external LTC source. That timecode becomes the output timecode for the first transcoded frame. Then the output timecode counts up with each successive frame in the entire output.

Option 2: ISO 8601 UTC time with no dashes or colons. For example, 20160102T030405.678Z. In this case, the start time for every overlay will be the UTC time.

Option 3: You can only use this while adding or modifying an overlay in a running event. Set this tag to an empty string to set the start time to “now”. With this option, the start time is never exact. You cannot use this option when creating an event or modifying a non-running event.

Example

The following request creates an event with the name myLiveEvent and includes a motion_image_inserter section that inserts the motion overlay logo.swf.

```xml
POST http://198.51.100.22/live_events

<?xml version="1.0" encoding="UTF-8"?>
<live_event>
    <name>myLiveEvent</name>
    ...
    <motion_image_inserter>
        <action_time>10:16:23:10</action_time>
        <active>true</active>
        <enable_rest>true</enable_rest>
        <full_frame>false</full_frame>
        <left>100</left>
        <top>200</top>
        <insertion_mode>swf</insertion_mode>
        <loop_input>true</loop_input>
        <motion_image_inserter_input>
            <uri>/data/logo.swf</uri>
            <swf_arguments>"variableA" : 15, "variableB" : 1;"</swf_arguments>
        </motion_image_inserter_input>
    </motion_image_inserter>
</live_event>
```

Modify SWF Motion Overlay on a Running Event

In a running event, modify the content or behavior of the existing SWF motion overlay.

Note

Commands sent during an event change the event XML. Therefore, if you export the XML and create a new event with it, the new event will have any overlays set up as they were set during the course of the event, not as they were when the event was created.

HTTP Request and Response

HTTP URL

POST http://<Live IP Address>/live_events/<live event id>/motion_image_inserter
AWS Elemental Live User Guide
XML Structure

Body of Request
XML content consisting of one motion_image_inserter element that contains the required tags. For the
tags, see the table in the section the section called “Create or Modify a Non-Running Event” (p. 182).
For information on the tags to include for diﬀerent actions, see the section called “Motion Graphic
Overlay” (p. 150). This information is in a table at the end of each section.
Response
The response repeats back the data that you posted with <ID> tags for motion_image_inserter and
motion_image_inserter_input.
Example
The following example request modiﬁes the running event with the ID 10. It changes the SWF arguments
for the ActionScript code on the SWF asset. (There is an assumption that the overlay is currently
running.)
POST http://198.51.100.22/live_events/10/motion_image_inserter
------------------------------------------------------------<motion_image_inserter>
<swf_arguments>{"variableA" : "15", "variableB" : "1"}</swf_arguments>
</motion_image_inserter>

XML Structure
Topics
• Static Overlay, Creating and Modifying a Non-Running Event (p. 196)
• Static Overlay: Modifying Overlay on a Running Event (p. 199)
• Motion Overlay: Creating and Modifying a Non-Running Event (p. 200)
• Motion Overlay: Modifying Overlay on a Running Event (p. 200)

Static Overlay, Creating and Modifying a Non-Running Event
Overlays at Top Level
<image_inserter>
enable_rest
<insertable_image>
duration
fade_in
fade_out
height
image_x
image_y
layer
Version 2.17
196


### Overlays in Input Section

#### Data in `<input>` Element

<table>
<thead>
<tr>
<th><code>&lt;input&gt;</code></th>
<th><code>&lt;image_inserter&gt;</code></th>
<th>enable_rest</th>
<th><code>&lt;insertable_image&gt;</code></th>
<th>duration</th>
<th>fade_in</th>
<th>fade_out</th>
<th>height</th>
<th>image_x</th>
<th>image_y</th>
<th>layer</th>
<th>opacity</th>
<th>start_time</th>
<th>width</th>
<th><code>&lt;image_inserter_input&gt;</code></th>
<th>certificate_file</th>
</tr>
</thead>
</table>

---

Version 2.17

197
## Overlays in Stream Assembly Section

### Data in `<stream_assembly>`

<table>
<thead>
<tr>
<th><code>&lt;stream_assembly&gt;</code></th>
<th><code>&lt;video_description&gt;</code></th>
<th><code>&lt;video_preprocessors&gt;</code></th>
<th><code>&lt;image_inserter&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>&lt;video_description&gt;</code></td>
<td><code>&lt;video_preprocessors&gt;</code></td>
<td><code>&lt;image_inserter&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>enable_rest</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>&lt;insertable_image&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>duration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>fade_in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>fade_out</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>height</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>image_x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>image_y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>opacity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>start_time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>width</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>&lt;image_inserter_input&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>certificate_file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>password</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>uri</td>
</tr>
</tbody>
</table>
### Static Overlay: Modifying Overlay on a Running Event

```
<image_inserter>
  <image>
    activate
duration
    fade_in
    fade_out
    height
    image_x
    image_y
    layer
    opacity
    start_time
    width
  <image_inserter_input>
    certificate_file
    interface
    password
    uri
    username
  </image_inserter_input>
  </image>
</image_inserter>
```
Motion Overlay: Creating and Modifying a Non-Running Event

The data is inserted at the top level of the event.

```
<motion_image_inserter>
  action_time
  active
  enable_rest
  duration
  framerate_denominator
  framerate_numerator
  full_frame
  image_x
  image_y
  insertion_mode
  loop_input
  <motion_image_inserter_input>
    uri
    password
    username
  </motion_image_inserter_input>
</motion_image_inserter>
```

Motion Overlay: Modifying Overlay on a Running Event

The XML is identical to the XML when creating an event.

```
<motion_image_inserter>
  action_time
  active
  enable_rest
  duration
  framerate_denominator
  framerate_numerator
</motion_image_inserter>
```
Motion Graphic Overlay with SWF

Topics
- Is SWF the Best Choice? (p. 201)
- Working with Adobe Flash and SWF Files (p. 201)
- How AWS Elemental Live Integrates with ActionScript (p. 203)
- Validating Your Animations with swf-check (p. 204)

Is SWF the Best Choice?

With Shockwave Flash (SWF) files, AWS Elemental Live must render each frame of animation in real time and place it on the corresponding video frame. SWF rendering is resource-intensive, so poorly optimized SWF assets may degrade video output quality.

If your motion graphic overlay does not rely on any of the ActionScript abilities of Flash—namely passing in arguments and performing squeezeback—then an SWF file may not be the best choice, especially at high resolutions and/or high framerates. The other supported formats for motion graphic overlay assets can be cached in their entirety at the beginning of the event, avoiding resource issues with rendering in real time.

For motion graphics that rely on ActionScript, follow the best practices outlined in this document to create SWF files that are optimized to render with the least impact on AWS Elemental Live system resources.

Working with Adobe Flash and SWF Files

Topics
- Minimize Your Flash Stage Size (p. 202)
Minimize Your Flash Stage Size

If your overlay only takes up a small portion of the screen (such as a lower-third or animated bug), size your Flash stage as small as possible to still contain the asset's size/motion/movement. Position it within the underlying video in AWS Elemental Live when setting up the event. For example, instead of putting a 40x40 animation on a 1280x720 Flash canvas, size your canvas down to 40x40 (or just slightly larger to capture any dropshadow, embossing, and such).

To specify the position of the animation, go to the Motion Image Inserter portion of the Global Processors section of the AWS Elemental Live event. Use the “Left” and “Top” fields to specify the offset, in pixels, from the top left corner of the screen.

Optimize Flash Stage for Input

For best quality, the resolution and framerate of the animation should match those of the underlying video. We particularly recommend against scaling down.

If the animation resolution and framerate cannot be changed, AWS Elemental Live automatically adjusts the animation, but the quality of the animation may suffer.

Optimize Your Assets

Be sure the assets in your Flash animation are optimized for the canvas. Don't embed a series of large images and scale them down to fit. Instead use images that are properly sized before bringing them into Flash.

Sandbox the SWF File

Test the SWF file in the sandbox environment before using it. In the sandbox testing environment, the SWF file must not reference other resources on the filesystem or network, which would slow down the rendering of the frames.

Do Not Embed Video

The AWS Elemental Live motion graphic overlay feature does not have the ability to decode video. Therefore, you cannot embed video into your Flash animation.

Be Wary of Simulating Video in Flash

One way of simulating video in Flash is to include a sequence of images in the SWF file. This mechanism is supported but not recommended. We strongly encourage use of the native Flash animation components as Flash is optimized for those. Fall back to using sequences of images only as a last resort and only for small subsections of the screen as Flash is more reliable in those situations.

If you do use a sequence of images, it is especially important to optimize your images—make sure they are small and simple, crop them down to the smallest possible area, and keep them scaled 1:1 with the image canvas.
Set Publish Target and Script

Set your Publish Settings so that Target is Flash Player 11.1 and Script is set to ActionScript 3.0. This is the only version that is supported.

How AWS Elemental Live Integrates with ActionScript

How Flash Arguments Are Passed to ActionScript

Variables can be passed to the Flash asset via the swf_arguments tag in the Elemental event. These variables are accessible in the ActionScript code via stage.loaderInfo.parameters.

How Squeezeback Is Implemented

On the ActionScript Side

Squeezeback commands must be set up in ActionScript using ExternalInterface.call. You could set up this call so that it is executed, for example, in an event listener to the ActionScript ENTER_FRAME event.

The signature of the call() method must contain only five parameters:

- squeezeback
- x
- y
- width
- height

These are the only parameters that AWS Elemental Live handles. Parsing of the parameters is case-sensitive.

Examples of Squeezeback

This example instructs AWS Elemental Live to perform squeezeback on the underlying video at coordinates 0,0 (which positions its upper left corner in the upper left corner of the screen) and to resize the underlying video to 120 x 60 pixels.

```html
ExternalInterface.call(
    "squeezeback",
    "x=0",
    "y=0",
    "width=120",
    "height=60";
)
```

This example positions the video it at coordinates 1040,560 and to resize the underlying video to 240 x 160 pixels. (This positions it in the bottom right corner of a 1280 x 720 frame.)

```html
ExternalInterface.call(
    "squeezeback",
    "x=1040",
    "y=560",
    "width=120",
    "height=60";
)
```
Note
The command must be entered exactly as shown above, with each parameter in quotes. The values for the position and size parameters must be integers. Parameters are case-sensitive.

On the AWS Elemental Live Side
With the first squeezeback command, AWS Elemental Live resizes and/or positions the underlying video frame. This size and position are permanent: the underlying video does not snap back with the next frame. You are free to use any sort of squeezeback you like, including: distorting aspect ratio, smoothly adjusting squeezeback coordinates over time, or suddenly snapping to new coordinates.

The maximum frequency for squeezeback commands is based on the framerate of the underlying video. If the framerate is 60 fps, then AWS Elemental Live can send a command every 1/60 second.

Validating Your Animations with swf-check
AWS Elemental Live includes a tool called swf-check. The tool can do the following.

- Check Flash version and frame render times for your animation.
- Compare frame render times to the Flash animation's stated framerate and flag if it detects slow frames.

You should run the tool on real hardware in two situations:

- When the AWS Elemental Live node is idle to determine whether the Flash animation can be rendered in real time.
- While the AWS Elemental Live node encodes an event that does not include motion graphics. This test determines whether the Flash animation can render within the available resources while video encoding is in progress.

An example of swf-check being run via the command line and the feedback that is displayed on the screen for a SWF file does not work well at runtime:

```
[elemental@hostname ~]$ swf-check -f 30 ~/my_animation.swf
Elemental SWF validation tool
./UnsuccessfulExample.swf:
  - 1920x1080 pixels
  - 24.000000 fps (max 0.042 sec/frame)
  - 130 frames
  - SWF version: 14
  - Compression: not compressed
Frame 1: status 0: dirty rects (0,0-1920,1080) (0,0-0,0) (0,0-0,0) (0,0-0,0)
  Duration: 0.0697
  !!!!! Render time too long !!!!!
Frame 2: status 0: dirty rects (0,0-1920,1080) (0,0-0,0) (0,0-0,0) (0,0-0,0)
  Duration: 0.0668
  !!!!! Render time too long !!!!!
Frame 3: status 0: dirty rects (0,0-1920,1080) (0,0-0,0) (0,0-0,0) (0,0-0,0)
  Duration: 0.0644
  !!!!! Render time too long !!!!!
```

An example of swf-check feedback for a SWF file that runs successfully:

```
Elemental SWF validation tool
```

Version 2.17
204
Output Locking

This feature is a way to guarantee that events on multiple nodes (appliances) can produce video outputs that are frame-accurate with each other. It can be used to support output failure redundancy or distributed encoding.

Topics
- General Information About Output Locking (p. 205)
- Setting Up the Events (p. 206)

General Information About Output Locking

How Output Locking Works

The event on one of the locked encoders always starts encoding earlier than the other event or events (even if it is only a couple of frames earlier). This event is the “leader”. When subsequent events start, each drops encoded frames until it gets to a segment boundary. It obtains the master event’s embedded timecode and then locks to that timecode to remain synchronized through the life of the event.

The events communicate with each other (over multicast or unicast) in order to share the data required to ensure that their outputs are locked.

Output Failure Redundancy

Two events on two different nodes are locked to each other so that they communicate as described above. These two events are set up to ingest the same input to perform the same processing (they have identical video encoding parameters). The outputs are locked to each other, which means the same frame has the same timecode in both events.

The events are set up to publish to the same downstream packager. The packager is receiving both outputs, but it uses only one of them. If that output becomes unavailable or unusable, the packager switches to the second output. The packager can use the timecode from one output to synchronize the frames from that output to the corresponding frames in the output of the other event.

You can use multicast addresses or a unicast address to set up output locking for output failure redundancy.
You can lock any number of pairs of events. For example, event A (on node 1) and event B (on node 2) as a redundant pair, events C (on node 1) and D (on node 2) as another redundant pair, and so on.

**Distributed Encoding**

One example of distributed encoding is in building an ABR stack. Two or more events (each event on a different node) have the same input but each event encodes a different part of the ABR stack: each event creates one or more outputs.

The events do not necessarily specify identical encoding. But the events are locked together so that they communicate as described above. The locking means that the same frame from the output or outputs in one event has the same timecode as the output or outputs in all the other events.

The events publish to the same downstream packager, where the ABR stack is put together. The packager can use the timecode from one output to synchronize the frame from the outputs from one event to the corresponding frame in the outputs from the other event.

You can use multicast addresses or a unicast address to set up output locking for distributed encoding:

- If you use multicast addresses, you can lock together any number of events. For example, event A, event B, and event C (probably each on a separate node) are all locked together and are communicating over multicast.
- If you use unicast addresses, you can lock together only two events.

**Input Requirements**

- Input type: serial digital interface (SDI) or transport stream or HTTP Live Streaming (HLS) (network only) source. File sources do not support output locking.
- Resolution: The inputs within one event must have the same resolution. The input for one event can have a different resolution to the input for another event.
- GOP: The inputs within one event and between two events must have the same Group of Pictures (GOP) structure and framerate.
- Input timecode: Required. An SDI source must have an embedded or Linear/Longitudinal Timecode (LTC) timecode. A transport stream source or HLS source must have an embedded timecode.

**Output Requirements**

- Output types: Microsoft Smooth output, UDP/TS output, or an MPEG-2 TS container in an Archive output. All video encodes must use H.264 or H.265.
- Framerate: Within one event, the framerate in and out must be the same. Between two events, the framerate can be different but one must be a whole-number multiple of the other.

**Setting Up the Events**

**Input Fields**

<table>
<thead>
<tr>
<th>Location in Event</th>
<th>Field Name</th>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input &gt; Video Selector</td>
<td>Timecode Source</td>
<td>For SDI inputs, choose <strong>Embedded</strong> or LTC.</td>
<td>AWS Elemental Live uses this timecode to lock the events.</td>
</tr>
</tbody>
</table>
### Global Processors Fields for Multicast

<table>
<thead>
<tr>
<th>Location in Event</th>
<th>Field Name</th>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Processors &gt; Output Locking</td>
<td>Multicast</td>
<td>Select this field.</td>
<td></td>
</tr>
<tr>
<td>Global Processors &gt; Output Locking</td>
<td>Address</td>
<td>Enter the multicast address of any server.</td>
<td>The events communicate with each other via this address.</td>
</tr>
<tr>
<td>Global Processors &gt; Output Locking</td>
<td>Port (optional)</td>
<td>Enter the identical address, port, and interface across all events you want to lock together.</td>
<td></td>
</tr>
<tr>
<td>Global Processors &gt; Output Locking</td>
<td>Interface (optional)</td>
<td>Enter the identical address, port, and interface across all events you want to lock together.</td>
<td></td>
</tr>
</tbody>
</table>

### Global Processors Fields for Unicast

<table>
<thead>
<tr>
<th>Location in Event</th>
<th>Field Name</th>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Processors &gt; Output Locking</td>
<td>Multicast</td>
<td>De-select this field.</td>
<td></td>
</tr>
<tr>
<td>Global Processors &gt; Output Locking</td>
<td>Send Address</td>
<td>Enter the address where the other event is listening.</td>
<td>This event sends a message to the other event via this address.</td>
</tr>
<tr>
<td>Global Processors &gt; Output Locking</td>
<td>Send Port (required)</td>
<td>Enter the address where the other event is listening.</td>
<td></td>
</tr>
<tr>
<td>Global Processors &gt; Output Locking</td>
<td>Send Interface (optional)</td>
<td>Enter the address where the other event is listening.</td>
<td></td>
</tr>
</tbody>
</table>
### Setting Up the Events

<table>
<thead>
<tr>
<th>Location in Event</th>
<th>Field Name</th>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Processors &gt; Output Locking</td>
<td>Receive Port (required)</td>
<td>Enter the address where this event is listening.</td>
<td>The other event sends message to this event via this address.</td>
</tr>
<tr>
<td></td>
<td>Receive Interface (optional)</td>
<td>See the tooltips on the AWS Elemental Live web interface for more information. Before locking together more than one pair of events, make sure each pair is assigned to a different port from the other pair.</td>
<td></td>
</tr>
</tbody>
</table>

### Archive Output Fields

<table>
<thead>
<tr>
<th>Location in Event</th>
<th>Field Name</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Groups &gt; Archive</td>
<td>Destination</td>
<td>Choose MPEG-2 Transport Stream.</td>
</tr>
<tr>
<td>Output Groups &gt; Archive &gt; Output</td>
<td>Container</td>
<td></td>
</tr>
</tbody>
</table>

### MSS Output Fields

<table>
<thead>
<tr>
<th>Location in Event</th>
<th>Field Name</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Groups &gt;</td>
<td>Custom Group Name</td>
<td>Enter the same name across all events.</td>
</tr>
<tr>
<td>Use Event ID</td>
<td>De-selected</td>
<td></td>
</tr>
<tr>
<td>Send EOS</td>
<td>De-selected</td>
<td></td>
</tr>
<tr>
<td>Send Delay</td>
<td>Complete as desired. See the tooltip on the AWS Elemental Live web interface for details.</td>
<td></td>
</tr>
<tr>
<td>Fragment length</td>
<td>This field can be different in each event.</td>
<td></td>
</tr>
</tbody>
</table>
Stream > Video Fields

<table>
<thead>
<tr>
<th>Location in Event</th>
<th>Field Name</th>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream &gt; Video</td>
<td>Codec</td>
<td>H.264 or H.265</td>
<td>Must be the same codec in all events.</td>
</tr>
<tr>
<td></td>
<td>Framerate</td>
<td>Ensure that the values you set in the events are whole-number multiples of each other.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Segmentation Markers</td>
<td>None or EBP CableLabs.</td>
<td></td>
</tr>
</tbody>
</table>

SCTE-35 and SCTE-104 Message Processing in AWS Elemental Live

You can use AWS Elemental Live to manipulate the SCTE-35 messages in a TS input and to manipulate the SCTE-104 messages in an HD-SDI input. You can also use AWS Elemental Live to remove or include the cueing information conveyed by SCTE messages in the output streams (video, audio, closed captioning, data) and any associated manifests.

You set up SCTE message processing instructions in the AWS Elemental Live event. This guide describes how to perform this set up.

Note that AWS Elemental Live does not support processing of manifests that are present in the input. The information in these manifests is not ingested by AWS Elemental Live and is not included in the output or the output manifest.

About this guide

SCTE messages may convey DPI cueing information for ad avails and for other non-ad-avail messages such as programs and chapters.

This guide covers both Event Signaling and Management (ESAM) and non-ESAM processing of messages.

Assumptions

You should be familiar with the SCTE-35 and SCTE-104 standards and optionally with the SCTE-67 standards and how the input you encode implements these standards. You should be familiar with profiles and with managing AWS Elemental Live events. To use the REST API features, you should be familiar with interacting with AWS Elemental Live through the API.

Topics

- Eligible Messages and Streams (p. 210)
- Getting Ready: Setting the Ad Avail Mode (p. 221)
- Manifest Decoration (p. 222)
- Ad Avail Blanking and Blackout (p. 227)
- Passthrough or Removal of SCTE Messages (p. 232)
- SCTE-35 Message Insertion into Currently Running Events (p. 234)
- POIS Conditioning (p. 247)
Eligible Messages and Streams

The following messages and streams are eligible for SCTE-35 and SCTE-104 message processing:

- SCTE-35 messages in MPEG-2 transport stream (TS) inputs. These messages may or may not include segmentation descriptors.
- SCTE-104 messages in HD-SDI stream inputs.

Note
SCTE-104 messages are converted to SCTE-35 messages early in the processing, so this guide uses “SCTE-35 messages” to refer to both SCTE-35 messages and SCTE-104 messages.

SCTE Message Processing Options

The Options

The processing possibilities for SCTE messages include the following.

Automatically Convert SCTE-104 Messages

SCTE-104 messages in the input are automatically converted to SCTE-35 messages during processing of the input. No setup is required for this processing.

Line Up SCTE Messages to Use Blanking and Blackout of Output Content

The "cue out" and "cue in" instructions in SCTE-35 messages line up with specific content in the video, audio, and closed captions streams. You can set up to blank out this content in the output.

- The content for ad avails is blanked out using the Ad avail blanking feature.
- The content for other messages is blanked out using the blackout feature.

Your desired behavior must be set up in the event or profile.

SCTE-35 Pass-through

You can include SCTE-35 messages in the output data stream in any TS output. Your desired behavior must be set up in the event or profile.

Manifest Decoration

You can set up SCTE messages with manifest decoration with these options:

- HLS and HDS outputs can be set up so that their manifests include instructions that correspond to the original SCTE-35 message content.
- MS Smooth outputs can be set up to include these instructions in the sparse track.
- RTMP outputs can be set up to include these instructions in the data stream (because RTMP outputs do not have manifests).

Conditioning by a POIS
Optionally, SCTE-35 messages (including those converted from SCTE-104) can be diverted to a POIS server for ESAM conditioning. This conditioning is in addition to all the other processing (manifest decoration, blanking and blackout, and passthrough).

POIS and ESAM conditioning are described in POIS Conditioning (p. 247).

**Insert SCTE-35 Messages into Content During an Event**

The AWS Elemental Live REST API includes commands to insert SCTE-35 messages into the content while the AWS Elemental Live event is running.

You can insert using the AWS Elemental Live REST API. AWS Elemental Live makes no distinction between these messages and those that were in the original input. So, for example, if manifest decoration is enabled and you insert a message during encoding, that new message is represented in the manifest.

**Default Behavior of SCTE-35 and SCTE-104 Messages**

The default handling of SCTE-35 and SCTE-104 messages in AWS Elemental Live includes the following:

- No manifest decoration: You do not convert any SCTE-35 messages to event information in any output manifests or data streams.
- No passthrough: You do not pass through SCTE-35 or SCTE-104 messages in any data stream outputs.
- No blanking: You do not blank out video content for any events: you leave the content as is.

If you desire the default behavior as described, you have the information you need for processing SCTE messages and do not need to read the remainder of this guide.

**About Timecode Configuration and Timers**

The event or profile includes a timecode configuration field that identifies the source for timecode stamps to be inserted in the output. The source for these stamps may be a timecode embedded in the input or may be a source external to the input (for example, the system clock or a specified time). The following graphic shows where the timecode configuration fields are in the event.

Before starting the transcode, the transcoder gets the timecode from the source.

After the initial sampling, AWS Elemental Live calculates the timecode of every frame and attaches it to the output. The timecode stops advancing if there is no output. So, for example, if the input fails at 10:55:03:009, the timecode at that point is 10:55:009. If the input restarts 3 seconds later, the timecode of the next frame may be 10:55:03:012. The timecode will not be 10:55:06:009.
Given the importance of accurate times with SCTE-35 messages, ensure that the **Network Time Protocol (NTP)** is configured on the node.

## Scope of Processing Depending on Outputs

The following table summarizes which options apply to which kind of output. Following the table are details for each output type.

<table>
<thead>
<tr>
<th>Output</th>
<th>Pass-through in TS Outputs</th>
<th>Manifest Decoration</th>
<th>Blanking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive outputs with MPEG-2 as the container</td>
<td>Include all the original SCTE-35 messages. Convert any SCTE-104 messages to SCTE-35 messages (of the same message type) and include in the TS output.</td>
<td>Not applicable</td>
<td>Applicable</td>
</tr>
<tr>
<td>Archive outputs with other containers</td>
<td>Not applicable</td>
<td>Not applicable</td>
<td>Applicable</td>
</tr>
<tr>
<td>HLS</td>
<td>Include all the original SCTE-35 messages. Convert any SCTE-104 messages to SCTE-35 messages (of the same message type) and include in the TS output. Note that, with HLS, you either implement both manifest decoration and passthrough or you implement neither.</td>
<td>Decorate the HLS manifest with one or more of the following types of ad markers: - Adobe - Elemental - SCTE-35 enhanced.</td>
<td>Applicable</td>
</tr>
<tr>
<td>DASH</td>
<td>Not applicable</td>
<td>Not applicable</td>
<td>Applicable</td>
</tr>
<tr>
<td>HDS</td>
<td>Not applicable</td>
<td>Decorate the HDS manifest with one or more of the following types of ad markers: - onCuePoint SCTE-35 - Primetime DPI SCTE-35 - Primetime DPI Simple.</td>
<td>Applicable</td>
</tr>
<tr>
<td>MS Smooth</td>
<td>Not applicable</td>
<td>Include information on the SCTE-35 event in the sparse track.</td>
<td>Applicable</td>
</tr>
</tbody>
</table>
### Eligible Messages and Streams

<table>
<thead>
<tr>
<th>Output</th>
<th>Pass-through in TS Outputs</th>
<th>Manifest Decoration</th>
<th>Blanking</th>
</tr>
</thead>
</table>
| RTMP     | Not applicable              | Include one or more of the following types of ad markers in the RTMP datastream:  
• OnAkamaiAdPod  
• OnCuePoint  
• OnCuePoint SCTE-35  
• OnUserDataEvent | Applicable         |
| UDP/TS   | Include all the original SCTE-35 messages.  
Convert any SCTE-104 messages to SCTE-35 messages (of the same message type) and include in the TS output. | Not applicable | Applicable |
| RTSP     | Not applicable              | Not applicable | Applicable |

#### Topics
- Archive Output with MPEG-2 Container (p. 213)
- Archive Output with Other Containers (p. 214)
- Apple HLS Output (p. 215)
- DASH Output (p. 216)
- Adobe HDS Output (p. 216)
- MS Smooth Output (p. 217)
- Adobe RTMP Output (p. 218)
- UDP/TS Output (p. 219)
- RTSP Output (p. 220)

### Archive Output with MPEG-2 Container

A transport stream (TS) in an MPEG-2 container supports passthrough of the SCTE-35 messages, but it does not support creation of a manifest. Therefore, usable options are:

<table>
<thead>
<tr>
<th>SCTE-35 Passthrough</th>
<th>Insertion of SCTE-35 Messages</th>
<th>Manifest Decoration</th>
<th>Blanking and Blackout</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Yes or No</td>
<td>Not applicable</td>
<td>Yes or No</td>
<td>Turns on passthrough of SCTE-35 messages. In this case, you could also insert</td>
</tr>
</tbody>
</table>
### Eligible Messages and Streams

<table>
<thead>
<tr>
<th>SCTE-35 Passthrough</th>
<th>Insertion of SCTE-35 Messages</th>
<th>Manifest Decoration</th>
<th>Blanking and Blackout</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>No</td>
<td>Not applicable</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Disabled

No

Not applicable

No

Turns off passthrough in order to remove SCTE-35 messages from the video stream. Do not insert extra messages: they simply get stripped out of the output. Do not implement blanking or blackout.

Choose this option only if, in a downstream system, you do not want to replace video that was originally marked by cues.

### Archive Output with Other Containers

Other archive outputs do not support passthrough of the SCTE-35 messages or manifest decoration. Therefore, the only workable option is the default behavior:

<table>
<thead>
<tr>
<th>SCTE-35 Passthrough</th>
<th>Insertion of SCTE-35 Messages</th>
<th>Manifest Decoration</th>
<th>Blanking and Blackout</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td>No</td>
<td>Not applicable</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Not applicable

No

Not applicable

No

Removes SCTE-35 messages from the output. The manifest is not decorated. Do not implement blanking or blackout because, without SCTE-35 messages in the video stream and without manifest decoration, the output will not contain the necessary information for downstream systems to properly interpret the video content.
Apple HLS Output

Apple HLS output supports both passthrough of the SCTE-35 messages and manifest decoration. In fact, with HLS outputs, passthrough and manifest decoration are either both enabled or both disabled. Therefore, workable options are:

<table>
<thead>
<tr>
<th>SCTE-35 Passthrough</th>
<th>Insertion of SCTE-35 Messages</th>
<th>Manifest Decoration</th>
<th>Blanking and Blackout</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Yes or No</td>
<td>Enabled</td>
<td>Yes or No</td>
<td>Turns on passthrough of SCTE-35 messages and manifest decoration. In this case, you could also insert more SCTE-35 message if desired. You could also implement blanking and blackout.</td>
</tr>
<tr>
<td>Disabled</td>
<td>No</td>
<td>Disabled</td>
<td>No</td>
<td>Turns off passthrough in order to remove SCTE-35 messages from the video stream. Turns off manifest decoration. Do not insert extra messages: they simply get stripped from the output. Do not implement blanking or blackout. Choose this option only if, in a downstream system, you do not want to replace</td>
</tr>
</tbody>
</table>
### DASH Output

DASH ISO output does not support passthrough of the SCTE-35 messages or manifest decoration. Therefore, the only workable option is the default behavior:

<table>
<thead>
<tr>
<th>SCTE-35 Passthrough</th>
<th>Insertion of SCTE-35 Messages</th>
<th>Manifest Decoration</th>
<th>Blanking and Blackout</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td>No</td>
<td>Not applicable</td>
<td>No</td>
<td>Removes SCTE-35 messages from the output. The manifest is not decorated. Do not implement blanking or blackout because, without SCTE-35 messages in the video stream and without manifest decoration, it is impossible to find these blanks and blackouts programmatically.</td>
</tr>
</tbody>
</table>

### Adobe HDS Output

Adobe HDS output does not support passthrough of the SCTE-35 messages but does support manifest decoration. Therefore, the options that are workable are:

<table>
<thead>
<tr>
<th>SCTE-35 Passthrough</th>
<th>Insertion of SCTE-35 Messages</th>
<th>Manifest Decoration</th>
<th>Blanking and Blackout</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td>Yes or No</td>
<td>Enabled</td>
<td>Yes or No</td>
<td>Removes SCTE-35 messages from the video stream, but includes instructions in the manifest. You could insert extra messages: although they are not included in the</td>
</tr>
<tr>
<td>SCTE-35 Passthrough</td>
<td>Insertion of SCTE-35 Messages</td>
<td>Manifest Decoration</td>
<td>Blanking and Blackout</td>
<td>Effect</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------</td>
<td>--------------------</td>
<td>----------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Not applicable</td>
<td>No</td>
<td>Disabled</td>
<td>No</td>
<td>Removes SCTE-35 messages from the output. The manifest is not decorated. Do not implement blanking or blackout because, without SCTE-35 messages in the video stream and without manifest decoration, it is impossible to find these blanks and blackouts programmatically.</td>
</tr>
</tbody>
</table>

**MS Smooth Output**

MSS output does not support passthrough of the SCTE-35 messages but does support instructions in the sparse track. Therefore, the workable options are:

<table>
<thead>
<tr>
<th>SCTE-35 Passthrough</th>
<th>Insertion of SCTE-35 Messages</th>
<th>Manifest Decoration</th>
<th>Blanking and Blackout</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td>Yes or No</td>
<td>Enabled</td>
<td>Yes or No</td>
<td>Removes SCTE-35 messages from the video stream. But instructions are included in the sparse track. You could insert extra messages: although they are not included in the video stream of the output, they are represented by instructions in the sparse</td>
</tr>
</tbody>
</table>
### SCTE-35 Passthrough

<table>
<thead>
<tr>
<th>SCTE-35 Passthrough</th>
<th>Insertion of SCTE-35 Messages</th>
<th>Manifest Decoration</th>
<th>Blanking and Blackout</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td>No</td>
<td>Disabled</td>
<td>No</td>
<td>Removes SCTE-35 messages from the output. The sparse track does not include instructions. Do not implement blanking or blackout because, without SCTE-35 messages in the video stream and, without data in the sparse track, it is impossible to find these blanks and blackouts programmatically.</td>
</tr>
</tbody>
</table>

### Adobe RTMP Output

Adobe RTMP output does not support passthrough of the SCTE-35 messages but does support manifest decoration. Therefore, the workable options are:

<table>
<thead>
<tr>
<th>SCTE-35 Passthrough</th>
<th>Insertion of SCTE-35 Messages</th>
<th>Manifest Decoration</th>
<th>Blanking and Blackout</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td>Yes or No</td>
<td>Enabled</td>
<td>Yes or No</td>
<td>Removes SCTE-35 messages from the video stream. But instructions are included in the manifest. You could insert extra messages: although they are not included in the video stream of the output, they are represented by instructions in the manifest. You could also implement blanking and blackout.</td>
</tr>
<tr>
<td>SCTE-35 Passthrough</td>
<td>Insertion of SCTE-35 Messages</td>
<td>Manifest Decoration</td>
<td>Blanking and Blackout</td>
<td>Effect</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------</td>
<td>---------------------</td>
<td>-----------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Not applicable</td>
<td>No</td>
<td>Disabled</td>
<td>No</td>
<td>Removes SCTE-35 messages from the output. The manifest is not decorated. Do not implement blanking or blackout because, without SCTE-35 messages in the video stream and without manifest decoration, it is impossible to find these blanks and blackouts programmatically.</td>
</tr>
</tbody>
</table>

**UDP/TS Output**

User Datagram Protocol (UDP)/TS output supports passthrough of the SCTE-35 messages, but it does not support creation of a manifest. Therefore, workable options are:

<table>
<thead>
<tr>
<th>SCTE-35 Passthrough</th>
<th>Insertion of SCTE-35 Messages</th>
<th>Manifest Decoration</th>
<th>Blanking and Blackout</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Yes or No</td>
<td>Not applicable</td>
<td>Yes or No</td>
<td>Turns on passthrough of SCTE-35 messages. In this case, you could also insert more SCTE-35 message if desired. You could also implement blanking and blackout.</td>
</tr>
<tr>
<td>Disabled</td>
<td>No</td>
<td>Not applicable</td>
<td>No</td>
<td>Turns off passthrough in order to remove SCTE-35 messages from the video stream. Do not insert extra messages: they are simply get stripped out of the output. Do not implement</td>
</tr>
</tbody>
</table>
Eligible Messages and Streams

SCTE-35 Passthrough | Insertion of SCTE-35 Messages | Manifest Decoration | Blanking and Blackout | Effect
--- | --- | --- | --- | ---
| | | | | blanking or blackout.
| | | | Choose this option only if, in a downstream system, you do not want to replace video that was originally marked by cues.

RTSP Output

Real-Time Streaming Protocol (RTSP) output does not support passthrough of the SCTE-35 messages or manifest decoration. Therefore, the only workable option is the default behavior:

| SCTE-35 Passthrough | Insertion of SCTE-35 Messages | Manifest Decoration | Blanking and Blackout | Effect |
| --- | --- | --- | --- | ---
| Not applicable | No | Not applicable | No | Removes SCTE-35 messages from the output. The manifest is not decorated. Do not implement blanking or blackout because, without SCTE-35 messages in the video stream and without manifest decoration, it is impossible to find these blanks and blackouts programmatically.

Blanking and Passthrough and Manifest Decoration

It is important to understand that the logic for blanking ad content works on the video content associated with the "ad avail event" while the logic for passthrough and manifest decoration works on the actual SCTE-35 message.

So you can blank ad avails and not pass through SCTE-35 messages or not blank ad avails and not pass through SCTE-35 messages and decorate the manifest or any combination: the actions are independent.

The only exception to this rule is for HLS outputs: manifest decoration and passthrough are either both enabled or both disabled.
Getting Ready: Setting the Ad Avail Mode

Read this section if you want to support any of the following features:

- Manifest decoration for all outputs.
- Ad avail blanking for all outputs. (Note that this section does not apply to the blackout feature.)

If you have several outputs and you want to do manifest decoration or ad avail blanking on some of the outputs and not on others, you have to set up two different profiles or events.

**Note**
You do not have to read this section if you are doing Blackout image insertion but you are not doing ad avail blanking or manifest decoration.

Set the Ad Avail Mode

You must set the Ad Avail mode. The Ad Avail mode applies to all outputs: it cannot be set uniquely for individual outputs. To set up the Ad Avail Mode, do the following.

1. In the Profile or Event screen, click Advanced Avail Controls (in the Input section towards the top of the screen):

2. In Ad Avail Trigger, choose the desired mode from the drop-down menu. This mode identifies which of all possible “ad avail” events are treated as “ad avails.” This distinction comes into play in manifest decoration and ad avail blanking. For more information, see Manifest Decoration (p. 222) and Ad Avail Blanking and Blackout (p. 227).

Typically, you select the mode to match the type ID that you already know the input is using to indicate “ad avail” events.

- **Splice Insert Mode**. Select this mode if the input uses splice inserts to indicate ad avails. The input may also contain messages for other events such as chapters or programs.

- **Time Signal with APOS Mode**. Select this mode if the input contains time signals of segmentation type placement opportunity. The input may also contain messages for other events such as chapters or programs.

The following table specifies how a message (containing a specific combination of message type and segmentation type) is treated depending on the mode that is specified. Each message is treated as either an ad avail, a non-ad avail, or as “other.” See the table below.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Message Type ID</th>
<th>Segmentation Type ID</th>
<th>Ad Avail Event</th>
<th>Not an Ad Avail Event</th>
<th>Other Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splice Insert Mode</td>
<td>Splice Insert</td>
<td>No segmentation descriptor present</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Manifest Decoration

You can choose to interpret SCTE-35 messages from the original input and insert corresponding instructions into the output manifest for the following outputs:

- HLS
- HDS
- MS Smooth (the instructions are inserted in the sparse track).
- RTMP (the instructions are inserted into the data stream, given that RTMP does not support manifests).

## How SCTE-35 Events Are Handled in Manifests

Based on your criteria for creating the event, you may insert the following information may be inserted into the manifest.
### Manifest Decoration

**Type of Instruction**

<table>
<thead>
<tr>
<th>Type of Instruction</th>
<th>When Inserted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base-64</td>
<td>Information about all SCTE-35 messages in the output is incorporated into the manifest; the entire SCTE-35 message is added in base-64 format.</td>
</tr>
<tr>
<td>Cue-out, Cue-in</td>
<td>SCTE-35 messages that are ad avails (see Getting Ready: Setting the Ad Avail Mode (p. 221)) result in the insertion of “cue-out, cue-in” instructions.</td>
</tr>
<tr>
<td>Blackout</td>
<td>Only applies to the SCTE-35 Enhanced ad marker style (for HLS output; see below). SCTE-35 messages that are not ad avails result in the insertion of “blackout start/end” instructions, assuming that blackout is enabled (Ad Avail Blanking and Blackout (p. 227)). If blackout is not enabled, these instructions are not inserted.</td>
</tr>
</tbody>
</table>

**Splice Insert Mode**

This table describes which of the three types of instructions from the table in the start of this section are inserted for each message type and segmentation type.

<table>
<thead>
<tr>
<th>Message Type ID</th>
<th>Segmentation Type ID</th>
<th>Base 64</th>
<th>Cue-out, Cue-in</th>
<th>Blackout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splice Insert</td>
<td>No segmentation descriptor present</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Provider advertisement</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Distributor advertisement</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Placement opportunity</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other type (e.g. Chapter, Program)</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Time Signal</td>
<td>Provider advertisement</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Distributor advertisement</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Placement opportunity</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other type (e.g. Chapter, Program)</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Time Signal APOS Mode**
This table describes which of the three types of instructions from the table in the start of this section are inserted for each message type/segmentation type. Note that many segmentation types are completely ignored: they do not get included in the manifest.

<table>
<thead>
<tr>
<th>Message Type Id</th>
<th>Segmentation Type ID</th>
<th>Base 64</th>
<th>Cue-out, Cue-in</th>
<th>Blackout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splice Insert</td>
<td>Any</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Signal</td>
<td>Provider advertisement</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Distributor advertisement</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Placement opportunity</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other type (e.g. Chapter, Program)</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Procedure to Enable Manifest Decoration**

**Apple HLS**

Enable manifest decoration at the output group level, which means that the manifests for all outputs in that group include instructions based on the SCTE-35 content.

In the Apple HLS output group section, open the Advanced section and complete the following fields:

- **Ad Markers**: Click to select a marker type. You can select more than one type.
See Example Manifests for Apple HLS (p. 253) for information about the different of markers.

The manifest for each output includes a separate set of tags for each type that you select.

**HDS**

Enable manifest decoration at the output group level, which means that the manifests for all outputs in that group include instructions based on the SCTE-35 content.

In the Adobe HDS output group section, open the Advanced section and complete the following fields:

- **Ad Markers**: Click to select a marker type. You can select more than one type.

The manifest for each output includes a separate set of tags for each type that you select.
MS Smooth

With MS Smooth, if you enable manifest decoration, you actually insert instructions in the sparse track.

Enable manifest decoration at the output group level, which means that the sparse tracks for all outputs in that group include instructions based on the SCTE-35 content.

In the MS Smooth output group section, complete the following fields:

- **Enable Sparse Track**: Click to select a marker type. You can select more than one type.
- **Acquisition Point ID**: Enter the address of the certificate, if encryption is enabled on the output.

Real Time Messaging Protocol (RTMP)

With RTMP, if you enable manifest decoration, you insert instructions in the data stream as RTMP does not support manifests.

Enable manifest decoration at the output group level, which means that the manifests for all outputs in that group include instructions based on the SCTE-35 content.

In the RTMP output group section, complete the following fields:

- **Ad Markers**: Click to select a marker type. You can select more than one type.

The data stream includes a separate set of tags for each type that you select.
Ad Avail Blanking and Blackout

You can turn on one or both of the following features to blank out the content associated with a
SCTE-35 event:

• "**Blackout**": Blank out the content for other types of SCTE-35 messages such as chapters and
  programs.
• "**Ad avail blanking**": Blank out the content for a SCTE-35 message that is considered an “ad
  avail” (according to the mode, Getting Ready: Setting the Ad Avail Mode (p. 221)).

In both features, the handling is one of the following.

• Replace the video content associated with the event with an image you specify or with a black image.
• Remove the audio associated with the event.
• Remove the closed captions associated with the event.

**Topics**

• Blanking is Global (p. 227)
• Scope of Blackout of SCTE-35 Messages (p. 227)
• Scope of Ad Avail Blanking of SCTE-35 Messages (p. 228)
• Procedure to Enable Ad Avail Blanking (p. 231)
• Procedure to Enable Blackout (p. 231)

**Blanking is Global**

Both Ad avail blanking and Blackout apply to all outputs. You cannot choose to blank out for some
outputs and not blank out for others: it is an all-or-nothing decision.

**Compare Blanking to Manifest Decoration and Passthrough**

Manifest decoration and passthrough have a smaller scope than blanking: they apply only to outputs
that support these features.

Take important note of this fact, because if you do not do passthrough and do not do manifest
decoration in a given output (because these are not supported or because you choose not to) but you do
implement blanking, there are no “markers” for where the blanked content occurs.

To identify where this blanking is occurring, look for the IDR I-frames that identify where the SCTE-35
message used to be.

**Scope of Blackout of SCTE-35 Messages**

All SCTE-35 messages that are “Other type” are blanked out as follows:

<table>
<thead>
<tr>
<th>SCTE-35 Segmentation Type</th>
<th>Blanking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programs</td>
<td>Always</td>
</tr>
<tr>
<td>Chapters</td>
<td>Always</td>
</tr>
<tr>
<td>Unscheduled events</td>
<td>Always</td>
</tr>
<tr>
<td>Network</td>
<td>See below.</td>
</tr>
</tbody>
</table>

Version 2.17
227
How Network End Blackout Differs from Other Events

Network end blackout is different from the other events that trigger a blackout because:

- With Network, blanking starts when the "Network End" instruction is encountered and ends when the "Network Start" instruction is encountered.
- With other events, blanking starts when the "event start" instruction is encountered and ends when the "event end" instruction is encountered.

Scope of Ad Avail Blanking of SCTE-35 Messages

For Ad avail blanking (but not for Blackout), the ad avail mode you set controls which SCTE-35 events result in blanking of the content.

Using Splice Insert Mode

This table describes which message type/segmentation type combination is blanked by Ad avail blanking when the Ad Avail mode (Getting Ready: Setting the Ad Avail Mode (p. 221)) is Splice Insert mode.

<table>
<thead>
<tr>
<th>Message Type ID</th>
<th>Segmentation Type ID</th>
<th>Blanked</th>
<th>Not Blanked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splice Insert</td>
<td>No segmentation descriptor present</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Provider advertisement</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Distributor advertisement</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Placement opportunity</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other type (e.g. Chapter, Program)</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Time Signal</td>
<td>Provider advertisement</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Distributor advertisement</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Placement opportunity</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other type (e.g. Chapter, Program)</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Using Time Signal APOS Mode

This table describes which message type/segmentation type combination is blanked out by Ad avail blanking when the Ad Avail mode (Getting Ready: Setting the Ad Avail Mode (p. 221)) is Time signal with APOS mode.

<table>
<thead>
<tr>
<th>Message Type ID</th>
<th>Segmentation Type ID</th>
<th>Blanked</th>
<th>Not blanked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splice Insert</td>
<td>Any</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Time Signal</td>
<td>Provider advertisement</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Version 2.17
228
Ad Avail Blanking and Restriction Flags: Restrictions in the Input

SCTE-35 messages of type time_signal always contain segmentation descriptors.

SCTE-35 messages of type splice_insert may or may not include segmentation descriptors.

If the input has SCTE-35 messages that do include segmentation descriptors, these segmentation descriptors always include two types of flags. These flags provide additional information as guidance for blanking in specific situations:

- **web_delivery_allowed_flag.**
  - "True" means that there is no restriction on including the ad avail event’s content in a stream intended for web delivery: you do not need to blank out content in streams intended for web delivery.
  - "False" means there is a restriction: you should blank out the content.

- **no_regional_blackout_flag.**
  - "True" means that there is no restriction on including the ad avail event’s video in a stream intended for regional markets: you do not need to blank out content in streams intended for regional markets.
  - "False" means there is a restriction: you should blank out the content.

If neither flag is present (usually the case with splice_inserts), then both are considered to be false: blanking should occur.

If both flags are present (which is usually the case; it is unusual to have only one flag present), then a “false” for one flag takes precedence over a “true” for the other flag: blanking should occur.

Typically, in any given message in the input only one of these flags would ever be set to “false”, so only one restriction would ever be in place. There would typically never be both a regional delivery restriction and a web delivery restriction. This is because, if content is considered restricted for regional delivery, then it would not also be considered restricted for web delivery (where the concept of a region does not relate).

To summarize, this table shows the blanking logic that applies to each ad avail event that is encountered:

<table>
<thead>
<tr>
<th>Message Type ID</th>
<th>Segmentation Type ID</th>
<th>Blanked</th>
<th>Not blanked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributor advertisement</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Placement opportunity</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other type (e.g. Chapter, Program)</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Ad Avail Blanking and Restriction Flags: Restrictions in the Input**

<table>
<thead>
<tr>
<th>Content of Corresponding SCTE-35 Message</th>
<th>Content of Corresponding SCTE-35 Message</th>
<th>Result</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web delivery allowed?</td>
<td>Regional delivery allowed?</td>
<td></td>
<td>This combination can occur only in a message</td>
</tr>
</tbody>
</table>
Ad Avail Blanking and Restriction Flags: AWS Elemental Live Handling of Restrictions

You can modify this default blanking behavior by instructing AWS Elemental Live to ignore a restriction flag that is set to "false", so that blanking will not occur for this ad avail event. In other words, to use this logic: “Even if the message indicates to blank content because a regional blackout is in place, do not follow this instruction. Ignore the fact that a regional blackout is in place and do not blank content.”

You modify the behavior by setting one or the other of the **Ignore** flags to:

- **Unchecked** (default). The logic as above applies: so continue to observe the restriction.
- **Checked**. Ignore the restriction and do not blank video for the ad avail event.

**Warning**
Never set both fields to **Ignore**!

To summarize, the Ignore flags make a difference only to the scenarios in this table:
If you selected Splice Insert as the Ad Avail mode, then you can assume that the SCTE-35 ad avail message not include the two restriction flags described above. Every SCTE-35 ad avail message should result in an ad avail.

Therefore, if you know that input contains splice inserts (not time signals), you should leave both AWS Elemental Live fields unchecked.

**Procedure to Enable Ad Avail Blanking**

**To enable ad avail blanking**

1. In the Profile or Event screen, click Advanced Avail Controls (in the Input section towards the top of the screen):

2. Check or uncheck the two restriction fields:
   - **Unchecked** (default): Observe the restriction and blank the content for the ad avail event.
   - **Checked**: Ignore the restriction and do not blank the content for the ad avail event.

3. Go down to the Global Processors section and complete the following fields:
   - **Ad Avail Blanking**: Click to turn on. The **Blanking Image** field appears.
   - **Blanking Image**: Specify a .bmp or .png file to use for the blanking. If you leave this field blank, a plain black image is inserted.

**Procedure to Enable Blackout**

**To enable blackout**

1. In the Profile or Event screen, go down to the Global Processors section and complete the following fields:
   - **Blackout Image Insertion**: Click to turn on. The Blanking Image field appears.
   - **Blanking Image**: Specify a .bmp or .png file to use for the blanking. If you leave this field empty, a plain black image is inserted.
2. If you want to enable network end blackout (in other words, blank content when network transmission has ended and remove blanking only when network transmission resumes), complete these fields:
   - **Enable Network End Blackout**: Checked.
   - **Network ID**: The Entertainment Identifier Registry (EIDR) ID of the network in the format 10.nnnn/xxxx-xxxx-xxxx-xxxx-xxxx-xxxx-c (case-insensitive). Only network end events with this ID trigger blackout.
   - **Network End Blackout Image**: Specify a .bmp or .png file to use for the blanking. If you leave this field blank, a plain black image is inserted.

### Passthrough or Removal of SCTE Messages

SCTE-35 messages from the input can be passed through (included) in the data stream for the following outputs.

- **Archive outputs with MPEG-2 as the container**:
  - **HLS**: You specify whether to pass through at the output group level: passthrough or removal applies globally to all outputs in the output group.
  - **UDP/TS**: You specify whether to pass through at the output level: for each individual output in the output group.

SCTE-104 messages are handled for each output as follows:

- If you choose to pass through the SCTE-35, then all SCTE-104 messages are converted to SCTE-35 messages (of the same message type) and included in the data stream.
- If you choose to remove the SCTE-35 messages, the SCTE-104 messages are also removed.
• Archive Procedure (p. 233)
• Apple HLS Passthrough Procedure (p. 233)
• UDP/TS (p. 234)

Archive Procedure

You enable or disable passthrough at the output level: only in outputs that have an MPEG-2 TS container.

1. In the Profile or Event screen, go to the Output Groups section at the bottom of the screen and display the tab for Archive Output Group.
2. In the output that has the MPEG-2 TS container, open the PID Control section. Complete the following fields:
   • SCTE-35: Click to check.
   • SCTE-35 PID: Enter the ID of the PID where you want the SCTE-35 messages to go.

Result

All SCTE-35 messages from the input are included in the data stream of this output.

Apple HLS Passthrough Procedure

Passthrough is enabled or disabled individually for each output, which means it can be applied differently for different outputs in the same group.

1. If you have not already set up for manifest decoration, do so now; see Procedure to Enable Manifest Decoration (p. 224).
2. In the Profile or Event screen, go to the Output Groups section at the bottom of the screen and display the tab for Apple HLS Output Group.
   a. In each output, open the PID Control section. You will note that the SCTE-35 field is automatically checked (because you set up for manifest decoration) and you cannot uncheck it.
   b. Complete the following field:
      • SCTE-35 PID field: Enter the ID of the PID where you want the SCTE-35 messages to go.
Result

All SCTE-35 messages from the input are included in the data stream of the relevant output.

UDP/TS

Passthrough is enabled or disabled individually for each output, which means it can be applied differently for different outputs in the same group.

To enable passthrough

1. In the Profile or Event screen, go to the Output Groups section at the bottom of the screen and display the tab for UDP/TS Output Group.
2. In the output where you want to pass through SCTE-35 messages, open the PID Control section. Complete the following fields:
   - SCTE-35: Click to check.
   - SCTE-35 PID: Enter the ID of the PID where you want the SCTE-35 messages to go.

Result

All SCTE-35 messages from the input are included in the data stream of the relevant output.

SCTE-35 Message Insertion into Currently Running Events

You can use the AWS Elemental Live REST API to insert SCTE-35 messages into an AWS Elemental Live event that is currently running. The API supports insertion of a SCTE-35 message of type splice_insert or time_signal (Working with Time Signals (p. 244)).
Working with Splice Inserts

Splice inserts inserted by the REST API are always of type “ad avail.”

You can:

• Insert a SCTE-35 ad avail of type splice_insert. See Insert a New Splice Insert Message (p. 235).
• Get the timecode of the content that is currently being processed. This data may be useful in determine the start time you need to enter in the command. See Get Current Time (p. 241).
• Insert an end time in an ad avail. See Insert an End Time in an Existing Ad Avail (p. 242).
• Cancel a SCTE-35 ad avail that is pending (the start time has not yet been reached). See Cancel a Pending Ad Avail (p. 243).

Effect of these commands

These commands modify the data stream as it is being encoded.

Their precise effect on other parts of the content depends on how the event is set up, as described in earlier chapters of this guide. So it depends on the Ad Avail mode, whether manifest decoration is enabled, blanking and blackout are enabled, and whether passthrough is enabled.

Insert a New Splice Insert Message

Inserts a SCTE-35 message of type splice_insert in the stream either immediately or at a specified time. The command always includes a start time. It may also include or omit a duration (which implies an end time).

The command does not support inclusion of a segmentation descriptor, which means that the message is always considered to be an “ad avail.”

HTTP URL

POST <IP address of Live node>/live_events/<ID of event>/cue_point

Body of HTTP

The XML body contains one cue_point elements containing the following tags:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Sub-tag</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_id</td>
<td></td>
<td>integer</td>
<td>Specify an ID for this SCTE-35 request to allow for canceling</td>
</tr>
</tbody>
</table>
### SCTE-35 Message Insertion into Currently Running Events

<table>
<thead>
<tr>
<th>Tag</th>
<th>Sub-tag</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>of the insertion later (Cancel a Pending Ad Avail (p. 243)).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Or leave blank, in which case an ID is generated and returned in the response.</td>
</tr>
<tr>
<td>splice_time</td>
<td></td>
<td></td>
<td>Include this in order to specify the insertion point relative to the stream timecode. Include either splice_time or splice_offset, not both.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See Specifying Time with splice_time Tag (p. 237) for details.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specify the time by including the hours, minutes, seconds, and frames tags.</td>
</tr>
<tr>
<td>hours</td>
<td></td>
<td>integer</td>
<td>The start time of the ad avail. All fields are required.</td>
</tr>
<tr>
<td>minutes</td>
<td></td>
<td>integer</td>
<td>Enter the time in 24-hour format.</td>
</tr>
<tr>
<td>seconds</td>
<td></td>
<td>integer</td>
<td>To insert the ad avail immediately (taking into account that there is a small delay while the request is processed), enter 0 in all fields.</td>
</tr>
<tr>
<td>frames</td>
<td></td>
<td>integer</td>
<td>The frame within the specified seconds at which to insert the ad avail.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If blank, the start time is the first frame in the specified second.</td>
</tr>
</tbody>
</table>
### Tag: `splice_offset`

<table>
<thead>
<tr>
<th>Sub-tag</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>integer</td>
<td>The start time of the ad avail. Include either <code>splice_time</code> or <code>splice_offset</code>, not both. Include in order to specify the start time for the ad avail as the specified milliseconds after the request is received. See Specifying Time with <code>splice_offset</code> Tag (p. 237). Specify the milliseconds. The number cannot be negative.</td>
</tr>
</tbody>
</table>

### Specifying Time with "splice_time" Tag

Use the `splice_offset` tag to specify the start time as a specific clock time, for example, at 10:20:33. The time you specify must match the timecode format in the event, as specified by the `Timecode Config Source` field in the event or profile. For example, if you specify 10:20:33 and the event uses system clock, the message is inserted at 10:20:33 UTC. If the event uses local time, the message is inserted at 10:20:33 for the time zone of the node.

To verify the timecode format in the event, submit a GET `live_events` request and (in the response) read the value in the `timecodeconfig` tag.

Splice Time requires either knowing in advance to insert an ad avail at a given time, or obtaining the current time and inserting an offset so that the time is not missed. It is easier to use Splice Time when you know start times in advance. You can obtain the timecode of the content currently being encoded; see Get Current Time (p. 241).

### Specifying Time with "splice_offset" Tag
You can use the `splice_offset` tag to specify time as a number of milliseconds into the future from the moment at which the command is executed. This offset may be 0, which means to insert immediately.

**Note**

Splice Offset is the recommended method if adding ad avails “on the fly” because it does not require knowing the current time of the video.

**Time of Insertion of the Message and Time for the Ad Avail**

The time the message is inserted and the start time for the ad avail are typically not identical. For example, you insert a message that says “insert an ad avail at 10:35:15:0”. The message is actually inserted in the content close to the moment that you enter the request. The message may actually be inserted at 10:35:00:0.

So you can insert the message in advance of its actual “targeted time.” In fact, it is a good idea to include some offset although take care when calculating offset.

- **Too little offset**: If you do not add enough offset, the specified time may have already passed. The ad avail may still be inserted in the video (at the current frame) but will be inserted too late for AWS Elemental Live to act upon it. For example, the command says “insert an ad avail at 10:35:15.0” but, if that time has passed, the message is inserted at 10:35:40.0 (for example). AWS Elemental Live inserts the message if its request time is less than 1 hour after the targeted time. Otherwise, it discards the message. Note that the message is inserted in the content but it has a start time that has already passed.

- **Too much offset**: The maximum offset is 6 hours. Within that range, the ad avail will be inserted at a timepoint (for example 10:35:20), and the command is “insert ad avail at 10:36:00.0, that is, in 40 seconds from now.”

**Duration of the Message and Time for the Ad Avail**

You can include a duration so that a start time is included and an end time is implied by the length of time for the duration. Or you can omit the duration so that only a start time is included. If you omit the duration, enter an end time separately.

**Response**

The body of the response is XML content consisting of one `response` element containing the following tags:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Sub-tag</th>
<th>Sub-sub-tag</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_id</td>
<td></td>
<td></td>
<td>integer</td>
<td>The event ID of this SCTE-35 request.</td>
</tr>
<tr>
<td>splice_time</td>
<td>hours</td>
<td></td>
<td>integer</td>
<td>If splice_time was specified, the hour, minutes, seconds and frame at which to insert the ad avail.</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td></td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>seconds</td>
<td></td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>frames</td>
<td></td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>splice_offset</td>
<td></td>
<td></td>
<td>integer</td>
<td>If splice_offset was specified, the time</td>
</tr>
</tbody>
</table>
## SCTE-35 Message Insertion into Currently Running Events

### Tag | Sub-tag | Sub-sub-tag | Type | Description
--- | --- | --- | --- | ---
message |  |  | string | A description of the action taken.
errors |  |  |  | Included only in an error response.
error | code |  |  | An error code.
error | message |  | string | A human-readable error message.

A success response does not include the `<errors>` element. A failure response contains only the `<event_id>` and `<errors>` elements.

### Example Message

The following shows an example message:

```xml
<message>
  Inserted event [32] at event time[08:02:38], PTS[00:02:20.982]. Avail time[08:02:38 0f] PTS[08:02:38.023], duration[01:00:00.000]. Current NTP [15:18:05.712]
</message>
```

### Data | Description
--- | ---
Inserted event [n] | The event ID for the ad avail request.
event time | The requested start time of the ad avail, as per the original request.

![Image of New Live Event Profile]

### PTS (first occurrence)
The time at which the SCTE-35 message insertion request was received by AWS Elemental Live, in a clock representation of the presentation timestamp (PTS). This PTS is a “timer”, not a clock time.

### Avail Time
The requested start time of the ad avail, including the frame. This time is in the timecode...
Specified in the event or profile. For more information, see About Timecode Configuration and Timers (p. 211).

This time is in the timecode specified in the event or profile. If the timecode configuration source is Clock time, Local time, and Specified time, this time is a “clock time.”

**PTS (second occurrence)**

The requested start time of the ad avail (with the frame converted to milliseconds).

**duration**

The duration of the ad avail, if specified, in 24-hour format.

**Current NTP**

The network time protocol (NTP) when the SCTE-35 message insertion request was received by AWS Elemental Live.

## Splice Insert Examples

### Splice Time

Insert a message into the event with the ID 3. Insert the message at 10 hours, 32 minutes, and 10 seconds, and give it a duration of 30 seconds. (The implied end time will be 10 hours, 32 minutes, and 40 seconds.)

```
POST 10.4.136.95/live_events/3/cue_point
----------------------------------------
<cue_point>
  <splice_time>
    <hours>10</hours>
    <minutes>32</minutes>
    <seconds>10</seconds>
  </splice_time>
  <duration>30</duration>
</cue_point>
```

The following shows a success response where **splice_offset** was used in the request. The SCTE-35 request has an ID of 8.

```
<response value="cue_point">
  <event_id>8</event_id>
  <splice_time>
    <hours>0</hours>
    <minutes>0</minutes>
    <seconds>0</seconds>
    <frames>0</frames>
  </splice_time>
  <splice_offset>8000</splice_offset>
  <message> Inserted at PTS[1234]. Avail time[00:00:05.000] PTS[2345], duration[30].</message>
</response>
```

The following shows a failure response:

```
<response value="cue_point">
```

Version 2.17

240
<event_id>8</event_id>
<errors>
  <error>
    <code>1040</code>
    <message>Preroll time must be positive integer</message>
  </error>
</errors>
</response>

Splice Offset

Insert a message into the event with the ID 3. Insert the message 8000 milliseconds from the moment the command is executed. The message has a duration of 30 seconds.

```xml
POST 10.4.136.95/live_events/3/cue_point

<cue_point>
  <splice_offset>8000</splice_offset>
  <duration>30</duration>
</cue_point>
```

Get Current Time

You can obtain the timecode of the frame that is currently being processed in the specified event. Refer to the following tables.

<table>
<thead>
<tr>
<th>HTTP URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST &lt;IP address of Live node&gt;/live_events/&lt;ID of event&gt;/cue_point</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Body of HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>The XML body contains one cue_point element containing the following tag:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_current_time</td>
<td>Integer</td>
<td>Always 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>The body of the response contains one response element containing the following tags:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Sub-tag</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td></td>
<td>integer</td>
<td>A unique ID that could be used in the event_id in a new request to insert a splice_insert. See the explanation below this table.</td>
</tr>
<tr>
<td>splice_time</td>
<td>hours</td>
<td>integer</td>
<td>The time (and frame) associated with the event at the moment that the API request was processed. Time</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>seconds</td>
<td>integer</td>
<td></td>
</tr>
</tbody>
</table>
SCTE-35 Message Insertion into Currently Running Events

<table>
<thead>
<tr>
<th>Tag</th>
<th>Sub-tag</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>frames</td>
<td>frames</td>
<td>integer</td>
<td>is in 24-hour format. For general information about timecodes in events, see About Timecode Configuration and Timers (p. 211).</td>
</tr>
<tr>
<td>message</td>
<td>message</td>
<td>string</td>
<td>A description of the time: the presentation timestamp (PTS) of the time and the time in NTP.</td>
</tr>
<tr>
<td>splice_offset</td>
<td>splice_offset</td>
<td>string</td>
<td>Always 0. See the explanation below this table.</td>
</tr>
<tr>
<td>value</td>
<td>value</td>
<td>string</td>
<td>Always “cue_point.”</td>
</tr>
</tbody>
</table>

The response is provided in this format so that you could take the entire response, clean it up a bit (for example, changing the "tag" tag to “event_id”), and use it as the body of a request to insert a spliceinsert (see Insert a New Splice Insert Message (p. 235)).

**Get Current Time Example**

The following shows a request for the current timecode in the event that has the ID 15:

```
POST 10.4.136.95/live_events/15/cue_point
-----------------------------------
<cue_point>
  <get_current_time>1</get_current_time>
</cue_point>
```

The following shows an example response for the request:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <tag>1</tag>
  <splice_time>
    <hours>0</hours>
    <minutes>0</minutes>
    <seconds>2</seconds>
    <frames>23</frames>
  </splice_time>
  <message>PTS[00:00:02.969]. Current NTP [16:21:23.573].</message>
  <splice_offset>0</splice_offset>
  <value>cue_point</value>
</response>
```

**Insert an End Time in an Existing Ad Avail**

You can insert an end time in the following situations:

- To add an end time to a message that has not yet started if an end time was not initially included.
- To cut short an ad avail that is in progress. The command can be used in this way if no end time was initially included or if it was included but you want to end early.
HTTP URL

PUT <IP address of Live node>/live_events/<ID of event>/cue_point

Body of HTTP

The XML body contains one cue_point element containing the following tags:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Sub-tag</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_id</td>
<td></td>
<td>integer</td>
<td>The event ID of the original POST cue_point (the request that did not include a duration).</td>
</tr>
<tr>
<td>return_offset</td>
<td></td>
<td>integer</td>
<td>The number of milliseconds to wait before inserting the ad avail.</td>
</tr>
</tbody>
</table>

Insert an End Time in an Existing Ad Avail Example

Insert an end time immediately into the event with the ID 4. The original SCTE-35 ad avail has an ID of 38.

PUT 10.4.136.96/live_events/4/cue_point
----------------------------------------
<cue_point>
  <event_id>38</event_id>
  <return_offset>0</return_offset>
</cue_point>

Cancel a Pending Ad Avail

If you inserted an ad avail the start time has not yet passed, you can cancel the insertion. No SCTE-35 message will be inserted.

HTTP URL

POST <IP address of Live node>/live_events/<ID of event>/cue_point

Body of HTTP

The XML body contains one cue_point element containing the following tag:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Sub-tag</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cancel_event_id</td>
<td></td>
<td>integer</td>
<td>The event ID of the original SCTE-35 request.</td>
</tr>
</tbody>
</table>
The response includes a message that identifies the event ID of the original request.

Cancel a Pending Ad Avail Example

The following shows a request to cancel the insertion of a pending ad avail in the event with the ID 4. The original SCTE-35 ad avail has an ID of 38:

```
POST 10.4.136.96/live_events/4/cue_point
----------------------------------------
<cue_point>
  <cancel_event_id>38</cancel_event_id>
</cue_point>
```

The following shows an example response for the request:

```
<response>
  <tag>13</tag>
  <event_id>5</event_id>
  <message>Canceled eventID [5]</message>
  <value>cue_point</value>
</response>
```

Working with Time Signals

Time signals inserted by the REST API can be one of the "ad avail" types, or some other type, depending on what you specify in the segmentation descriptor in the request.

Effect of Creating a Time Signal

Insertion of a time signal modifies the data stream as it is being encoded.

The effect of time signals on other parts of the content depends on how the event is set up, as described in earlier chapters of this guide. So it depends on the Ad Avail mode, whether manifest decoration is enabled, blanking and blackout are enabled, and whether passthrough is enabled.

Insert a New Time Signal Message

Inserts a SCTE-35 message of type time_signal in the stream either immediately or at a specified time. The command always includes a start time (in the time tag) and a duration (included in the segmentation descriptor).

HTTP URL

```
POST <IP address of Live node>/live_events/<ID of event>/time_signal
```

Body of HTTP
The XML body contains one `time_signal` element containing the following tags:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Sub-tag</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>hours</td>
<td>integer</td>
<td>The start time. All fields are required.</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td></td>
<td>Enter the time in 24-hour format.</td>
</tr>
<tr>
<td></td>
<td>seconds</td>
<td></td>
<td>To insert the message immediately (taking into account that there will be a small delay while the request is processed), enter 0 in all fields.</td>
</tr>
<tr>
<td>descriptors</td>
<td></td>
<td>string</td>
<td>Optional. A hexadecimal string containing time signal descriptor data in the form of a binary blob. See the SCTE-35 specification for details on the contents and format. This descriptor includes the time signal duration, segmentation type ID, and other key information.</td>
</tr>
</tbody>
</table>

Response

The body of the response is XML content consisting of one `response` element containing the following tags:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Sub-tag</th>
<th>Sub-sub-tag</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td></td>
<td></td>
<td>integer</td>
<td>A unique ID</td>
</tr>
<tr>
<td>signal_time</td>
<td>hours</td>
<td></td>
<td>integer</td>
<td>The start time, repeated from the request.</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td></td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>seconds</td>
<td></td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>frames</td>
<td></td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>message</td>
<td></td>
<td></td>
<td>string</td>
<td>A description of the action taken.</td>
</tr>
<tr>
<td>errors</td>
<td></td>
<td></td>
<td></td>
<td>Included only in an error response.</td>
</tr>
<tr>
<td></td>
<td>error</td>
<td>code</td>
<td>An error code.</td>
<td></td>
</tr>
</tbody>
</table>
### Tag

<table>
<thead>
<tr>
<th>Tag</th>
<th>Sub-tag</th>
<th>Sub-sub-tag</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>message</td>
<td>string</td>
<td></td>
<td>A human-readable error message.</td>
</tr>
</tbody>
</table>

A success response does not include the `<errors>` element. A failure response contains only the `<tag>` and `<errors>` elements.

### Example Message

The following shows an example message:

```
<message>Inserted event [32] at event time[08:02:38], PTS[00:02:20.982]. Avail time[08:02:38 0f] PTS[08:02:38.023], duration[01:00:00.000]. Current NTP [15:18:05.712]<message>
```

<table>
<thead>
<tr>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inserted event [n]</td>
<td>The event ID for the ad avail request.</td>
</tr>
<tr>
<td>event time</td>
<td>The requested start time of the ad avail, as per the original request.</td>
</tr>
<tr>
<td>PTS (first occurrence)</td>
<td>The time at which the SCTE-35 message insertion request was received by AWS Elemental Live, in a clock representation of the presentation timestamp (PTS). This PTS is a “timer”, not a clock time.</td>
</tr>
</tbody>
</table>
| Avail Time    | The requested start time of the ad avail, including the frame. This time is in the timecode specified in the event or profile.  
This time is in the timecode specified in the event or profile. If timecode configuration source is Clock time, Local time, and Specified time, this time is a “clock time.” |
| PTS (second occurrence) | The requested start time of the ad avail (with the frame converted to milliseconds). |
| duration      | The duration of the ad avail, if specified, in 24-hour format.               |
Data | Description
--- | ---
Current NTP | The network time protocol (NTP) when the SCTE-35 message insertion request was received by AWS Elemental Live.

**Insert a New Time Signal Message Example**

The following shows a request to insert a SCTE-35 message immediately into the event that has ID 3.

```xml
POST 10.4.136.95/live_events/3/time_signal

<time_signal>
  <time>
    <hours>0</hours>
    <minutes>0</minutes>
    <seconds>0</seconds>
    <frames>0</frames>
  </time>
  <descriptors>021B43554549000000027FBF030C54564E413130303030303031300000
  </descriptors>
</time_signal>
```

The following shows an example success response for the request:

```xml
<response value="time_signal">
  <message> Inserted time signal at event time[1234], PTS[1234]. Signal time[1234] PTS[1234]. </message>
  <tag>1</tag>
  <signal_time>
    <hours>0</hours>
    <minutes>0</minutes>
    <seconds>0</seconds>
  </signal_time>
</response>
```

The following shows an example failure response for the request:

```xml
<response value="time_signal">
  <tag>1</tag>
  <errors>
    <error>
      <code>1040</code>
      <message>Invalid time signal message</message>
    </error>
  </errors>
</response>
```

**POIS Conditioning**

AWS Elemental Live events can be configured to communicate with a POIS server. During processing of the input, each time a SCTE-35 message is encountered, AWS Elemental Live sends the message contents to the server. The POIS responds with SCTE-35 content that may be identical to, slightly different from, or completely different from the original.
AWS Elemental Live also supports handling of “out-of-band” SCTE-35 messages from the POIS – messages that are not a response to a message originally sent by AWS Elemental Live. If such a message is received, AWS Elemental Live accepts and processes it.

**Effect of POIS Conditioning**

When you enable POIS conditioning, an extra step is inserted into the regular processing of the SCTE-35 and SCTE-104 messages. The event’s ad avail mode, of manifest decoration, of ad avail blanking, of blackout and of SCTE-35 passthrough still apply to some degree.

Read the earlier sections of this guide, then read the following to determine how POIS conditioning changes the standard behavior.

**Ad Avail Mode**

When POIS conditioning is enabled, the ad avail mode is always set to “splice insert.” For information about the implications for manifest decoration and ad avail blanking that is performed by AWS Elemental Live, see Getting Ready: Setting the Ad Avail Mode (p. 221).

**SCTE-35 Messages Inserted by REST API**

All these messages are sent to the POIS along with messages that are already in the input.

**New or Conditioned SCTE-35 Messages**

The POIS can send an instruction to insert a new SCTE-35 message, modify an existing one, or delete an existing one. A message received from the POIS may include any message type and segmentation type.

**Manifest Decoration**

The effect of POIS conditioning on manifest decoration is as follows:

- **HLS Outputs** – A message received from the POIS may include instructions to decorate an HLS manifest (not other types of manifests). This information is used to decorate the HLS manifest.

  How AWS Elemental Live processes the decoration information depends on how the event or event has been configured:
  - If the event or event does not have manifest decoration enabled for HLS outputs, then the information is ignored.
  - If the event does have it enabled, then the decoration information is inserted in the manifest.
  - The information is inserted into the manifest “as is.” The style of the information may not match the styles (ad marker styles) specified in the event. AWS Elemental Live does not check for format inconsistencies between these instructions and the ad marker style.

- **Other Outputs** – Decoration of other manifest types is according to the information in the SCTE-35 message and how the AWS Elemental Live event is set up for manifest decoration and which ad avail mode is enabled. The POIS conditioning has no effect on these manifest types.

**Blanking and Blackout**

The effect of POIS conditioning on blanking and blackout is as follows:

- **Extra Blackout Instructions** – A message received from the POIS may include explicit “blank the content corresponding to this SCTE-35 message” instructions for any SCTE-35 message.

  Even if blanking or blackout (whichever applies to the message type and segmentation descriptor type) is disabled in AWS Elemental Live, AWS Elemental Live observes the instruction for this specific message.
• **Blanking Image** – A POIS response may include reference to a blanking .png or .bmp file. AWS Elemental Live uses this file for any blanking/blackout if it can find the file at /data/server/esam/ on the AWS Elemental Live node.

If AWS Elemental Live cannot find the file, it uses a black slate.

• **Restriction Flags** – The Override restriction flags in the AWS Elemental Live event (Ignore Web Delivery Allowed flag and Ignore No Regional Blackout flag) are always set to unchecked in ESAM Mode. See Ad Avail Blanking and Blackout (p. 227).

• **Passthrough or Removal** – If passthrough is enabled in the AWS Elemental Live event, then the rule is that all SCTE-35 message are passed through.

But when POIS conditioning is enabled, the POIS can override this rule: if the POIS instruction is to remove a given SCTE-35 message, then that message is removed and is not passed through, even though passthrough is enabled in AWS Elemental Live.

**Procedure to Enable POIS Conditioning**

1. In the Profile or Event screen, click Advanced Avail Controls (in the Input section towards the top of the screen):


3. Complete the fields as follows:
   a. Complete the first 6 fields to identify the endpoints on the POIS.
   b. For **Response Signal Preroll**, change the value as desired to set the distance (in milliseconds) between the time that the SCTE-35 message is inserted and the start time of that ad avail.
Setting Up via the REST API

This topic lists the parameters found on the AWS Elemental Live event or profile and specifies the location of those parameters in the XML for an event or profile. This topic does not cover control of SCTE-35 message via the REST API at runtime; that information is in SCTE-35 Message Insertion into Currently Running Events (p. 234).

### Set the Ad Avail Mode

<table>
<thead>
<tr>
<th>Field</th>
<th>XML Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger</td>
<td>ad_trigger</td>
</tr>
</tbody>
</table>

### Manifest Decoration

<table>
<thead>
<tr>
<th>Field</th>
<th>XML Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Group &gt; Apple HLS &gt; Advanced &gt; Ad Markers</td>
<td>output_group/apple_live_group_settings/ad_markers</td>
</tr>
<tr>
<td>Output Group &gt; Adobe HDS &gt; Advanced &gt; Ad Markers</td>
<td>output_group/hds_group_settings/ad_signaling</td>
</tr>
<tr>
<td>Output Group &gt; MS Smooth &gt; Enable Sparse Track</td>
<td>output_group/ms_smooth_group_settings/enable_sparse_track</td>
</tr>
<tr>
<td>Output Group &gt; MS Smooth &gt; Acquisition Point ID</td>
<td>output_group/ms_smooth_group_settings/acquisition_point_id</td>
</tr>
<tr>
<td>Output Group &gt; RTMP &gt; Ad Markers</td>
<td>output_group/rtmp_group_settings/ad_markers</td>
</tr>
</tbody>
</table>

### Ad Avail Blanking and Blackout

<table>
<thead>
<tr>
<th>Field</th>
<th>XML Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Avail Controls &gt; Ignore no_regional_blackout_flag</td>
<td>ignore_no_regional_blackout_flag</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ignore web_delivery_allowed_flag</td>
<td>ignore_web_delivery_allowed_flag</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Ad Avail Blanking &gt; On/Off</td>
<td>avail_blanking/enabled/</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Ad Avail Blanking &gt; Browse</td>
<td>avail_blanking/avail_blanking_image/certificate_file</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Ad Avail Blanking &gt; Browse</td>
<td>avail_blanking/avail_blanking_image/interface</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Ad Avail Blanking &gt; Credentials icon &gt; Password</td>
<td>avail_blanking/avail_blanking_image/password</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Ad Avail Blanking &gt; Browse</td>
<td>avail_blanking/avail_blanking_image/uri</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Ad Avail Blanking &gt; &gt; Credentials icon &gt; Username</td>
<td>avail_blanking/avail_blanking_image/username</td>
</tr>
<tr>
<td>Field</td>
<td>XML Tag</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Blackout Image Insertion &gt; On/Off</td>
<td>blackout_slate/enabled/</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Blackout Image Insertion &gt; Enable Network End Blackout &gt; Network ID</td>
<td>blackout_slate/network_id</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Blackout Image Insertion &gt; Browse</td>
<td>blackout_slate/blackout_slate_image/</td>
</tr>
<tr>
<td></td>
<td>certificate_file</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Blackout Image Insertion &gt; Browse</td>
<td>blackout_slate/blackout_slate_image/</td>
</tr>
<tr>
<td></td>
<td>interface</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Blackout Image Insertion &gt; Browse</td>
<td>blackout_slate/blackout_slate_image/</td>
</tr>
<tr>
<td></td>
<td>password</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Blackout Image Insertion &gt; Browse</td>
<td>blackout_slate/blackout_slate_image/</td>
</tr>
<tr>
<td></td>
<td>uri</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Blackout Image Insertion &gt; Browse</td>
<td>blackout_slate/blackout_slate_image/</td>
</tr>
<tr>
<td></td>
<td>username</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Blackout Image Insertion &gt; Enable Network End Blackout &gt; Network End Blackout Image &gt; Browse</td>
<td>blackout_slate/network_end_blackout_image/</td>
</tr>
<tr>
<td></td>
<td>certificate_file</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Blackout Image Insertion &gt; Enable Network End Blackout &gt; Network End Blackout Image &gt; Browse</td>
<td>blackout_slate/network_end_blackout_image/</td>
</tr>
<tr>
<td></td>
<td>interface</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Blackout Image Insertion &gt; Enable Network End Blackout &gt; Network End Blackout Image &gt; Credentials &gt; Password</td>
<td>blackout_slate/network_end_blackout_image/</td>
</tr>
<tr>
<td></td>
<td>password</td>
</tr>
<tr>
<td>Processors &gt; Global Processors &gt; Blackout Image Insertion &gt; Enable Network End Blackout &gt; Network End Blackout Image &gt; Credentials &gt; Username</td>
<td>blackout_slate/network_end_blackout_image/</td>
</tr>
<tr>
<td></td>
<td>username</td>
</tr>
</tbody>
</table>

**Passthrough or Removal**

<table>
<thead>
<tr>
<th>Field</th>
<th>XML Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive Output Group &gt; Output &gt; MPEG-2 TS &gt; PID Control &gt; SCTE-35</td>
<td>output_group/output/scte35Passthrough</td>
</tr>
<tr>
<td>Archive Output Group &gt; Output &gt; MPEG-2 TS &gt; PID Control &gt; SCTE-35 PID</td>
<td>output_group/output/m2ts_settings/scte35_pid</td>
</tr>
<tr>
<td>Apple HLS Output Group &gt; Output &gt; PID Control &gt; SCTE-35</td>
<td>output_group/output/scte35Passthrough</td>
</tr>
<tr>
<td>Field</td>
<td>XML Tag</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>Apple HLS Output Group &gt; Output &gt; PID Control &gt; SCTE-35 PID</td>
<td>output_group/output/m3u8_settings/scte35_pid</td>
</tr>
<tr>
<td>UDP/TS Output Group &gt; Output &gt; SCTE-35</td>
<td>output_group/output/scte35_passthrough</td>
</tr>
<tr>
<td>UDP/TS Output Group &gt; Output &gt; SCTE-35 PID</td>
<td>output_group/output/ts_settings/scte35_pid</td>
</tr>
</tbody>
</table>

**POIS Conditioning**

<table>
<thead>
<tr>
<th>Field</th>
<th>XML Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Acquisition Point Identifier</td>
<td>esam/acquisition_point_id/</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Asset URI Identifier</td>
<td>esam/asset_uri_id/</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Signal Conditioner Endpoint</td>
<td>esam/scc_uri/certificate_file</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Signal Conditioner Endpoint</td>
<td>esam/scc_uri/interface</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Signal Conditioner Endpoint</td>
<td>esam/scc_uri/password</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Signal Conditioner Endpoint</td>
<td>esam/scc_uri/uri</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Signal Conditioner Endpoint</td>
<td>esam/scc_uri/username</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Alternate Signal Conditioner Endpoint</td>
<td>esam/alternate_scc_uri/certificate_file</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Alternate Signal Conditioner Endpoint</td>
<td>esam/alternate_scc_uri/interface</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Alternate Signal Conditioner Endpoint</td>
<td>esam/alternate_scc_uri/password</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Alternate Signal Conditioner Endpoint</td>
<td>esam/alternate_scc_uri/uri</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Alternate Signal Conditioner Endpoint</td>
<td>esam/alternate_scc_uri/username</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Manifest Conditioner Endpoint</td>
<td>esam/mcc_uri/certificate_file</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Manifest Conditioner Endpoint</td>
<td>esam/mcc_uri/interface</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Manifest Conditioner Endpoint</td>
<td>esam/mcc_uri/password</td>
</tr>
<tr>
<td>Advanced Avail Controls &gt; Ad Avail Trigger &gt; Manifest Conditioner Endpoint</td>
<td>esam/mcc_uri/uri</td>
</tr>
</tbody>
</table>
Example Manifests for Apple HLS

This section lists example manifests for the Apple HLS manifest styles that AWS Elemental supports.

Example Manifest for Adobe Ad Marker

This does not insert any CUE-OUT CONT (continuation tags) to indicate to a player joining mid-break that there is a current avail. This does not insert a CUE-IN tag at the end of the avail.

Structure

<table>
<thead>
<tr>
<th>Segment</th>
<th>Tag</th>
<th>Tag Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment in which the ad avail starts.</td>
<td>1 CUE: DURATION tag</td>
<td>1</td>
</tr>
</tbody>
</table>

Tag Contents

- CUE:DURATION containing:
  - duration: Duration in fractional seconds.
  - id: An identifier, unique among all ad avails CUE tags.
  - type: SpliceOut
  - time: The PTS time for the ad avail, in fractional seconds.

Example

The following is the tag for an ad avail lasting 414.171 PTS.

```text
#EXT-X-CUE:DURATION="201.467",ID="0",TYPE="SpliceOut",TIME="414.171"
```
Example Manifest for AWS Elemental Ad Marker

Structure

<table>
<thead>
<tr>
<th>Segment</th>
<th>Tag</th>
<th>Tag Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment in which the ad avail starts</td>
<td>CUE-OUT</td>
<td>1</td>
</tr>
<tr>
<td>Each succeeding segment</td>
<td>CUE-OUT-CONT</td>
<td>0-n</td>
</tr>
<tr>
<td>Segment in which ad avail ends</td>
<td>CUE-IN</td>
<td>1</td>
</tr>
</tbody>
</table>

Tag Contents

- CUE-OUT contains Duration.
- CUE-OUT-CONT contains Elapsed time and Duration.
- CUE-IN has no content.

Example

```
#EXT-X-CUE-OUT:30.000
.
.
.
#EXT-X-CUE-OUT-CONT:8.308/30
.
.
.
#EXT-X-CUE-OUT-CONT:20.391/30
.
.
.
#EXT-X-CUE-IN
```

Example Manifest for SCTE-35 Enhanced Ad Marker

Structure

<table>
<thead>
<tr>
<th>Segment</th>
<th>Tag</th>
<th>Tag Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment in which the ad avail starts</td>
<td>OATCLS-SCTE35</td>
<td>1</td>
</tr>
<tr>
<td>Segment in which the ad avail starts</td>
<td>ASSET</td>
<td>1</td>
</tr>
<tr>
<td>Segment in which the ad avail starts</td>
<td>CUE-OUT</td>
<td>1</td>
</tr>
<tr>
<td>Each succeeding segment</td>
<td>CUE-OUT-CONT</td>
<td>0-n</td>
</tr>
<tr>
<td>Segment in which ad avail ends</td>
<td>CUE-IN</td>
<td>1</td>
</tr>
</tbody>
</table>

Tag Contents
AWS Elemental Live User Guide
SCTE-35 Ad Marker EXT-X-DATERANGE

- OATCLS-SCTE35 containing the base-64-encoded raw bytes of the original SCTE-35 ad avail message.
- ASSET containing the CAID or UPID as specified in the original SCTE35 message.
- 1 CUE-OUT per ad avail.
- CUE-OUT-CONT containing:
  - The elapsed time of the avail.
  - The duration declared in the original SCTE35 message.
  - SCTE35 containing the base-64-encoded raw bytes of the original SCTE-35 ad avail message.

These lines repeat until the ad avail ends.
- CUE-IN to indicate the end of the avail.

Example

```
#EXT-OATCLS-SCTE35:/DA0AAAAAABQb+ADAQ6QaehxDVUVJQAAA03/PAAEUrEoIICCMcmgABQb+ADAQ6QaehxDVUVJQAAA03/PAAEUrEoIICCMcmg
#EXT-X-ASSET:CAID=0x0000000020FB6501
#EXT-X-CUE-OUT:201.467
.
.
.
#EXT-X-CUE-OUT-CONT:ElapsedTime=5.939,Duration=201.467,SCTE35=/DA0AAAAAABQb+ADAQ6QaehxDVUVJQAAA03/PAAEUrEoIICCMcmgABQb+ADAQ6QaehxDVUVJQAAA03/PAAEUrEoIICCMcmg
.
.
.
#EXT-X-CUE-IN
```

**SCTE-35 Ad Marker EXT-X-DATERANGE**

This section is an addendum to the section called “SCTE-35 and SCTE-104 Message Processing” (p. 209). AWS Elemental Live supports the EXT-X-DATERANGE Ad Marker style in the manifest created for an HLS output.

To select this marker, do the following:

- In the event, go to Output Group > Apple HLS > Advanced > Ad Markers and select EXT-X-DATERANGE.

The structure and contents of the line are as described in HTTP Live Streaming, draft-pantos-http-live-streaming-23 (Version 23).

This marker style applies when the SCTE-35 messages are time_signal or splice_insert, and when the messages are passed through from the input or inserted at runtime using the REST API.

**Example**

This example shows an ad avail with an ID of 999 and a planned duration of 30 seconds. The first EXT-X-DATERANGE, for the start of the ad avail, includes a START-DATE. The second EXT-X-DATERANGE includes an END-DATE and a time that is 30.03 seconds later.

**Note**

The DURATION, PLANNED-DURATION and END-DATE tags are optional. If they are not present in the input or not specified in the REST API command, then these tags are omitted from the manifest.

```
#EXT-X-DATERANGE:ID="999",START-DATE="2018-08-22T21:54:00.079Z",PLANNED-DURATION=30.000,
```
Ingesting from SMPTE-2110

AWS Elemental Live supports ingest of SMPTE-2110 video, audio, and ancillary data using a 10 Gb or higher Ethernet network card installed on your appliance.

**Important**

- AWS Elemental Live expects that all stream inputs have an SDP file that describes them, and that they are available via HTTP or HTTPS.
- All SMPTE-2110 inputs must include a video stream and an audio stream.
- AWS Elemental Live reads the SDP file only when an input starts. Changes to the SMPTE-2110 feeds are not supported while an input is running.

Supported Sources

**Video**

AWS Elemental Live supports SMPTE-2110 video with the characteristics in the following table.

<table>
<thead>
<tr>
<th>Resolutions</th>
<th>Framerate</th>
<th>Video Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD up to 1080i and 1080p</td>
<td>25–60 fps</td>
<td>4:2:2 10-bit YUV source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AWS Elemental Live supports integer and non-integer framerates.</td>
</tr>
</tbody>
</table>

**Audio**

AWS Elemental Live supports one uncompressed PCM audio stream with up to 16 channels. The audio stream must have sampling rates of 48,000 or 44,100.

**Captions**

The caption source can contain any number of ancillary streams that contain 608, 708, or teletext captions.

**Other Ancillary Data**

AWS Elemental Live supports SCTE-104 ad avails and Active Format Description (AFD).

**Note**

AWS Elemental Live does not currently support ingest of LTC timecodes in ancillary data.
## Setting Up a New Event

**To set up a new event**

1. In **Name**, enter a name for your event.
2. In **Priority**, enter a priority number for the event.
3. Choose ** Restart on Failure** if you want the node to restart when it recovers from a failure.
4. For **Input**, select **SMPTE 2110 Input**.
5. (Optional) In the associated text field, replace *(name)* with a name for your input.
6. In **Video SDP Location**, enter a URL for the video.
7. In **Media Index** and **Interface**, enter information for the video as described here:
   - For **Media Index**, enter the index of the stream. This is the location of this particular stream within the SDP file. For example, enter 0 if it's the first stream, 1 if it's the second stream, and so on.
   - **Note**
     Each SDP file describes one or more video streams, audio streams, and/or ancillary data streams in no particular order. You must have at least one SDP file that contains a video stream.
   - For **Interface**, enter the network interface to use for this particular stream. For example, *eth2*.
8. If needed, repeat step 6 and step 7 for the audio and ancillary data streams.
9. Set up the remainder of the event as described in the AWS Elemental Live API Reference.

## Event Processing

When you start the event, AWS Elemental Live locates the SDP file during the stream discovery process. AWS Elemental Live reads the SDP file only one time during the input startup, so ensure that your source characteristics don't change.

**Note**
AWS Elemental Live does not currently use NMOS specifications to discover or manage devices.

## Reference: Supported Captions

This document is intended to ensure that when creating an event or profile, you select a format that is valid for your output captions.

Various constraints exist for the caption formats that you can include in your content:

- For a given input container, AWS Elemental Live can parse certain caption formats.
- For a given input caption format, AWS Elemental Live can create one or more output captions. However, a given output caption can appear only in specific output containers.

In summary – starting with the output container you want to produce:

- A given output container supports a given set of output caption formats. This container constrains your choice of output captions.
- Furthermore, to produce each output caption format, you must use one of the compatible input caption formats. And you must select an input caption format which can appear in the input container you have selected. So both which original input container and format you choose constrains your choice of output caption formats.
## Supported Caption Formats

<table>
<thead>
<tr>
<th>Caption</th>
<th>Supported in Input</th>
<th>Supported in Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ancillary data</td>
<td>✓</td>
<td></td>
<td>• From MXF input, data that is compliant with “SMPTE 291M: Ancillary Data Package and Space Formatting” and that is contained in ancillary data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• From QuickTime® input or for QuickTime output, data that is compliant with EIA-608 (also known as CEA-608) or CEA-708 (also known as EIA-708) and that is contained in ancillary data.</td>
</tr>
<tr>
<td>Ancillary+Embedded</td>
<td></td>
<td>✓</td>
<td>For QuickTime output only, the output combines captions in ancillary data and embedded captions. The ancillary captions are compliant with EIA-608 (also known as CEA-608) or CEA-708 (also known as EIA-708). The embedded captions are described later in this table.</td>
</tr>
<tr>
<td>ARIB</td>
<td>✓</td>
<td>✓</td>
<td>Captions that are compliant with the ARIB STD-B37 Version 2.4.</td>
</tr>
<tr>
<td>Burn-in</td>
<td>N/A</td>
<td>✓</td>
<td>From input: It is technically impossible for AWS Elemental Live to read burn-in captions. Therefore, from an input viewpoint, they cannot</td>
</tr>
<tr>
<td>Caption</td>
<td>Supported in Input</td>
<td>Supported in Output</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------</td>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CFF-TT</td>
<td></td>
<td>✓</td>
<td>Captions for Ultraviolet output.</td>
</tr>
<tr>
<td>DVB-Sub</td>
<td>✓</td>
<td>✓</td>
<td>Captions that are compliant with ETSI EN 300 743.</td>
</tr>
<tr>
<td>EBU-TT-D</td>
<td></td>
<td>✓</td>
<td>Captions that are compliant with EBU Tech 3380, EBU-TT-D Subtitling Distribution Format, 2018.</td>
</tr>
<tr>
<td>Embedded</td>
<td>✓</td>
<td>✓</td>
<td>Captions that are compliant with the EIA-608 standard (also known as CEA-608 or SMPTE-259M or “line 21 captions”) or the CEA-708 standard (also known as EIA-708).</td>
</tr>
<tr>
<td>Embedded+SCTE-20</td>
<td>✓</td>
<td>✓</td>
<td>Captions that have both embedded and SCTE-20 in the video. The embedded captions are inserted before the SCTE-20 captions.</td>
</tr>
<tr>
<td>RTMP CaptionInfo</td>
<td></td>
<td>✓</td>
<td>Captions that are compliant with the Adobe onCaptionInfo format.</td>
</tr>
<tr>
<td>RTMP CuePoint</td>
<td></td>
<td>✓</td>
<td>Captions that are in the cuePoint format.</td>
</tr>
<tr>
<td>SCC</td>
<td>✓</td>
<td>✓</td>
<td>Captions that are in the Scenarist format, file extension .scc.</td>
</tr>
<tr>
<td>Caption</td>
<td>Supported in Input</td>
<td>Supported in Output</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------</td>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SCTE-20</td>
<td>✓</td>
<td></td>
<td>Captions that are compliant with the standard “SCTE 20 2012 Methods for Carriage of CEA-608 Closed Captions and Non-Real Time Sampled Video.”</td>
</tr>
<tr>
<td>SCTE-20+Embedded</td>
<td></td>
<td>✓</td>
<td>Captions that are compliant with SCTE-43. The SCTE-20 captions are inserted in the video before the Embedded captions.</td>
</tr>
<tr>
<td>SCTE-27</td>
<td>✓</td>
<td></td>
<td>Captions that are compliant with the standard “SCTE-27 (2011), Subtitling Methods for Broadcast Cable.”</td>
</tr>
<tr>
<td>SMI</td>
<td>✓</td>
<td>✓</td>
<td>Captions in the Microsoft SAMI format.</td>
</tr>
<tr>
<td>SMPTE-TT</td>
<td></td>
<td>✓</td>
<td>Captions that are compliant with the standard “SMPTE ST 2052-1:2010.”</td>
</tr>
<tr>
<td>SRT</td>
<td>✓</td>
<td>✓</td>
<td>Captions in the SRT format.</td>
</tr>
<tr>
<td>STL</td>
<td>✓</td>
<td></td>
<td>Captions in the EBU STL format. Spruce STL format is not supported.</td>
</tr>
</tbody>
</table>
### Supported Caption Formats

<table>
<thead>
<tr>
<th>Caption</th>
<th>Supported in Input</th>
<th>Supported in Output</th>
<th>Description</th>
</tr>
</thead>
</table>
| Teletext | ✓                  | ✓                   | From SDI input. Captions in:  
  
  - OP42 teletext format. SMPTE 2031 field is unchecked in source.  
  - OP47 teletext format wrapped in a SMPTE-2031 envelope. SMPTE 2031 field is checked in source.  
  - OP47 teletext format, also known as SMPTE RDD-08 (compliant with ITU-R BT.1120-7). SMPTE 2031 field is unchecked in source.  
  
  From TS input: Captions in the EBU Teletext format.  
  
  From MXF file input:  
  OP47 teletext format, also known as SMPTE RDD-08 (compliant with ITU-R BT.1120-7). SMPTE 2031 field is unchecked in source.  
  
  For output: Captions in the EBU Teletext format. |
| TTML     | ✓                  | ✓                   | Caption files that are compliant with the standard “Timed Text Markup Language 1 (TTML1) (Second Edition).” |
| WebVTT   |                    | ✓                   | Captions that are compliant with “webvtt: The Web Video Text Tracks Format” [webvtt](http://dev.w3.org/html5/webvtt/). |
Supported Output Containers

All output containers can include captions except the XDCAM container. The only way to include captions with output that is in an XDCAM container is to create a captions-only container; see the section called “Captions-only Output Container” (p. 276).

Support Tables

Topics
• GPP Output Container (p. 262)
• DASH Output Container (p. 263)
• HDS or MP4 Output Container (p. 264)
• HLS Output Container (p. 265)
• HLS fMP4 Output Container (p. 266)
• MPEG2-TS File Output Container or MPEG2-UDP Streaming Output Container (p. 267)
• MSS Output Container (p. 268)
• MXF Output Container (p. 270)
• QuickTime Output Container (p. 271)
• Raw Output Container (p. 272)
• RTMP Output Container (p. 273)
• RTSP Output Container (p. 274)
• Ultraviolet Output Container (p. 275)
• Captions-only Output Container (p. 276)

GPP Output Container

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.

<table>
<thead>
<tr>
<th>Source Input Container</th>
<th>Source Caption Format</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td></td>
</tr>
<tr>
<td>MP4 Container</td>
<td>Embedded</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td></td>
</tr>
<tr>
<td>MXF Container</td>
<td>Embedded</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in</td>
</tr>
<tr>
<td>QuickTime Container</td>
<td>Embedded</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in</td>
</tr>
<tr>
<td>Raw Container</td>
<td>SRT</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SMI</td>
<td>Burn-in</td>
</tr>
</tbody>
</table>
### Source Input Container

<table>
<thead>
<tr>
<th>Source Input Container</th>
<th>Source Caption Format</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TTML</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>STL</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SCC</td>
<td>Burn-in</td>
</tr>
<tr>
<td>RTMP Container</td>
<td>Embedded</td>
<td>Burn-in</td>
</tr>
<tr>
<td>SDI Container</td>
<td>Embedded</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>ARIB</td>
<td>None</td>
</tr>
<tr>
<td>MPEG2-TS Container</td>
<td>Embedded</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>ARIB</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>DVB-Sub</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SCTE-27</td>
<td>Burn-in</td>
</tr>
</tbody>
</table>

### DASH Output Container

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>MP4 Container</td>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>MXF Container</td>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>QuickTime Container</td>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
</tbody>
</table>
### Source Caption Container | Source Caption Input | Supported Output Captions
---|---|---
Ancillary Data | | Burn-in, SMPTE-TT, TTML, EBU-TT-D

### Raw Container

<table>
<thead>
<tr>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRT</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>SMI</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>TTML</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>STL</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>SCC</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
</tbody>
</table>

### RTMP Container

<table>
<thead>
<tr>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
</tbody>
</table>

### SDI Container

<table>
<thead>
<tr>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>Teletext</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>ARIB</td>
<td>None</td>
</tr>
</tbody>
</table>

### MPEG2-TS Container

<table>
<thead>
<tr>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>SCTE-20</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>Teletext</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>ARIB</td>
<td>None</td>
</tr>
<tr>
<td>DVB-Sub</td>
<td>Burn-in, SMPTE-TT</td>
</tr>
<tr>
<td>SCTE-27</td>
<td>Burn-in, SMPTE-TT</td>
</tr>
</tbody>
</table>

### HDS or MP4 Output Container

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded</td>
</tr>
</tbody>
</table>

### MP4 Container

<table>
<thead>
<tr>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
</tbody>
</table>
### HLS Output Container

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, Web-VTT</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded, Web-VTT</td>
</tr>
<tr>
<td>MP4 Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, Web-VTT</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded, Web-VTT</td>
</tr>
<tr>
<td>MXF Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, Web-VTT</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, Embedded, Web-VTT</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>None</td>
</tr>
</tbody>
</table>
### Source Caption Container | Source Caption Input | Supported Output Captions
--- | --- | ---
QuickTime Container | Embedded | Burn-in, Embedded, Web-VTT
| Ancillary Data | Burn-in, Embedded, Web-VTT
Raw Container | SRT | Burn-in, Web-VTT
| SMI | Burn-in, Web-VTT
| TTML | Burn-in, Web-VTT
| STL | Burn-in, Web-VTT
| SCC | Burn-in, Embedded, Web-VTT
RTMP Container | Embedded | Burn-in, Embedded, Web-VTT
SDI Container | Embedded | Burn-in, Embedded, Web-VTT
| Teletext | Burn-in, Web-VTT
| ARIB | None
MPEG2-TS Container | Embedded | Burn-in, Embedded, Web-VTT
| SCTE-20 | Burn-in, Embedded, Web-VTT
| Teletext | Burn-in, Web-VTT
| ARIB | None
| DVB-Sub | Burn-in
| SCTE-27 | Burn-in

### HLS fMP4 Output Container

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.

### Source Caption Container | Source Caption Input | Supported Output Captions
--- | --- | ---
HLS Container | Embedded | Web-VTT
| SCTE-20 | Web-VTT
MP4 Container | Embedded | Web-VTT
| SCTE-20 | Web-VTT
MXF Container | Embedded | Web-VTT
| Ancillary Data | Web-VTT
QuickTime Container | Embedded | Web-VTT
| Ancillary Data | Web-VTT
Raw Container | SRT | Web-VTT
### Source Caption Container | Source Caption Input | Supported Output Captions
--- | --- | ---
SMI |  | Web-VTT
TTML |  | Web-VTT
STL |  | Web-VTT
SCC |  | Web-VTT
RTMP Container | Embedded | Web-VTT
SDI Container | Embedded | Web-VTT
| Teletext | Web-VTT
MPEG2-TS Container | Embedded | Web-VTT
| SCTE-20 | Web-VTT
| Teletext | Web-VTT

**MPEG2-TS File Output Container or MPEG2-UDP Streaming Output Container**

To read this table, find the type of container and captions from your input. The supported caption formats for this *output* container are then shown in the last column.

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td>MP4 Container</td>
<td>Embedded</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td>MXF Container</td>
<td>Embedded</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in, DVB-Sub, Teletext</td>
</tr>
<tr>
<td>Source Caption Container</td>
<td>Source Caption Input</td>
<td>Supported Output Captions</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>QuickTime Container</td>
<td>Embedded</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td>Ancillary Data</td>
<td></td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td>Raw Container</td>
<td>SRT</td>
<td>Burn-in, DVB-Sub</td>
</tr>
<tr>
<td></td>
<td>SMI</td>
<td>Burn-in, DVB-Sub</td>
</tr>
<tr>
<td></td>
<td>TTML</td>
<td>Burn-in, DVB-Sub</td>
</tr>
<tr>
<td></td>
<td>STL</td>
<td>Burn-in, DVB-Sub</td>
</tr>
<tr>
<td></td>
<td>SCC</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td>RTMP Container</td>
<td>Embedded</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td>SDI Container</td>
<td>Embedded</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in, DVB-Sub, Teletext</td>
</tr>
<tr>
<td></td>
<td>ARIB</td>
<td>ARIB</td>
</tr>
<tr>
<td>MPEG2-TS Container</td>
<td>Embedded</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, DVB-Sub, Embedded, Embedded+SCTE-20, SCTE-20+Embedded</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in, DVB-Sub, Teletext</td>
</tr>
<tr>
<td></td>
<td>ARIB</td>
<td>ARIB</td>
</tr>
<tr>
<td></td>
<td>DVB-Sub</td>
<td>Burn-in, DVB-Sub</td>
</tr>
<tr>
<td></td>
<td>SCTE-27</td>
<td>Burn-in, DVB-Sub</td>
</tr>
</tbody>
</table>

**MSS Output Container**

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.
<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>MP4 Container</td>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>MXF Container</td>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>QuickTime Container</td>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>Raw Container</td>
<td>SRT</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>SMI</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>TTML</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>STL</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>SCC</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>RTMP Container</td>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td>SDI Container</td>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>ARIB</td>
<td>None</td>
</tr>
<tr>
<td>MPEG2-TS Container</td>
<td>Embedded</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, SMPTE-TT, TTML, EBU-TT-D</td>
</tr>
</tbody>
</table>
**MXF Output Container**

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td>SCTE-20</td>
<td>Burn-in</td>
<td>Embedded</td>
</tr>
<tr>
<td>MP4 Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td>MXF Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in</td>
</tr>
<tr>
<td>QuickTime Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td>Ancillary Data</td>
<td>Burn-in</td>
<td>Embedded</td>
</tr>
<tr>
<td>Raw Container</td>
<td>SRT</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SMI</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>TTML</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>STL</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SCC</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td>RTMP Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td>SDI Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>ARIB</td>
<td>None</td>
</tr>
<tr>
<td>MPEG2-TS Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in</td>
</tr>
</tbody>
</table>
## QuickTime Output Container

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td>MP4 Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td>MXF Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in</td>
</tr>
<tr>
<td>QuickTime Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td>Raw Container</td>
<td>SRT</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SMI</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>TTML</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>STL</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SCC</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td>RTMP Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td>SDI Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in</td>
</tr>
</tbody>
</table>
### Support Tables

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIB</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MPEG2-TS Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded, Embedded +Ancillary</td>
</tr>
<tr>
<td>Teletext</td>
<td></td>
<td>Burn-in</td>
</tr>
<tr>
<td>ARIB</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>DVB-Sub</td>
<td>Burn-in</td>
<td></td>
</tr>
<tr>
<td>SCTE-27</td>
<td>Burn-in</td>
<td></td>
</tr>
</tbody>
</table>

**Raw Output Container**

This table describes the caption formats that can be included in a raw output container that contains video. For information on support when the captions are in a raw container on their own (independent of video), see Captions-only Output Container (p. 276).

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td>MP4 Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td>MXF Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in</td>
</tr>
<tr>
<td>QuickTime Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td>Raw Container</td>
<td>SRT</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SMI</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>TTML</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>STL</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SCC</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td>RTMP Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
<tr>
<td>SDI Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded</td>
</tr>
</tbody>
</table>
### RTMP Output Container

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td>MP4 Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td>MXF Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in, RTMP CuePoint</td>
</tr>
<tr>
<td>QuickTime Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td>Raw Container</td>
<td>SRT</td>
<td>Burn-in, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>SMI</td>
<td>Burn-in, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>TTML</td>
<td>Burn-in, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>STL</td>
<td>Burn-in, RTMP CuePoint</td>
</tr>
<tr>
<td>Source Caption Container</td>
<td>Source Caption Input</td>
<td>Supported Output Captions</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>SCC</td>
<td></td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td>RTMP Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td>SDI Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>ARIB</td>
<td></td>
</tr>
<tr>
<td>MPEG2-TS Container</td>
<td>Embedded</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, Embedded, RTMP CaptionInfo, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in, RTMP CuePoint</td>
</tr>
<tr>
<td></td>
<td>ARIB</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>DVB-Sub</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SCTE-27</td>
<td>Burn-in</td>
</tr>
</tbody>
</table>

**RTSP Output Container**

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in</td>
</tr>
<tr>
<td>MP4 Container</td>
<td>Embedded</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in</td>
</tr>
<tr>
<td>MXF Container</td>
<td>Embedded</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in</td>
</tr>
<tr>
<td>QuickTime Container</td>
<td>Embedded</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in</td>
</tr>
<tr>
<td>Raw Container</td>
<td>SRT</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>SMI</td>
<td>Burn-in</td>
</tr>
<tr>
<td></td>
<td>TTML</td>
<td>Burn-in</td>
</tr>
</tbody>
</table>
## Support Tables

### Ultraviolet Output Container

To read this table, find the type of container and captions from your input. The supported caption formats for this output container are then shown in the last column.

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Source Caption Input</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS Container</td>
<td>Embedded</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td>MP4 Container</td>
<td>Embedded</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td></td>
<td>SCTE-20</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td>MXF Container</td>
<td>Embedded</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td></td>
<td>Teletext</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td>QuickTime Container</td>
<td>Embedded</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td></td>
<td>Ancillary Data</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td>Raw Container</td>
<td>SRT</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td></td>
<td>SMI</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td></td>
<td>TTML</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td></td>
<td>STL</td>
<td>Burn-in, CFF-TT</td>
</tr>
<tr>
<td></td>
<td>SCC</td>
<td>Burn-in, CFF-TT</td>
</tr>
</tbody>
</table>
## Captions-only Output Container

This table describes the caption formats that can be included on their own in an output. With this option, the container is always a raw container that contains only the captions (video would be in another container that may be a raw container or may be some other type).

If you have one of the source caption formats listed in the first column – regardless of the source container – you can convert it to an external captions file and include it in a raw container that contains only that captions file.

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Original Caption Format</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any container</td>
<td>SRT</td>
<td>SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
</tr>
<tr>
<td></td>
<td>SMI</td>
<td>SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
</tr>
<tr>
<td></td>
<td>TTML</td>
<td>SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
</tr>
<tr>
<td></td>
<td>SMPTE-TT</td>
<td>SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
</tr>
<tr>
<td></td>
<td>STL</td>
<td>SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
</tr>
<tr>
<td>Embedded</td>
<td></td>
<td>SCC, SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
</tr>
<tr>
<td>SCC</td>
<td></td>
<td>SCC, SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
</tr>
<tr>
<td>SCTE-20</td>
<td></td>
<td>SCC, SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
</tr>
</tbody>
</table>
**Reference: Supported Codecs and Containers**

**Topics**
- General Information on Codecs and Containers (p. 277)
- Containers and Codecs by Input/Output (p. 282)

**General Information on Codecs and Containers**

**Topics**
- Definition of Supported Codecs and Containers (p. 277)
- H.264 (AVC) Support (p. 279)
- HEVC (H.265) Support (p. 279)
- MPEG-2 (H.262) Support (p. 280)
- Codec Support for CPU-Only and GPU-Enabled Deployments (p. 280)
- Transcode Support and Passthrough Support for Dolby Audio Codecs (p. 281)
- Rules for Ingesting Apple HLS TS Sources (p. 282)
- Rules for Ingesting MPEG-TS Programs (p. 282)

**Definition of Supported Codecs and Containers**

<table>
<thead>
<tr>
<th>Codec or Container</th>
<th>Direction</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>MXF input container for video</td>
<td>Input</td>
<td>Complete list of supported containers is: AS-02; OP-1a, OP-1b, OP-1c, OP-2a, OP-2b, and OP-2c.</td>
</tr>
<tr>
<td>Apple ProRes video codec</td>
<td>Input</td>
<td>Complete list of supported codecs is: Apple Prores 444 (all profiles), Apple Prores 4444 (all profiles), and Apple Prores 422 (all profiles). Apple Prores</td>
</tr>
</tbody>
</table>

**Reference: Supported Codecs and Containers**

<table>
<thead>
<tr>
<th>Source Caption Container</th>
<th>Original Caption Format</th>
<th>Supported Output Captions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCTE-20+Embedded</td>
<td>SCC, SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
<td></td>
</tr>
<tr>
<td>Embedded+SCTE-20</td>
<td>SCC, SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
<td></td>
</tr>
<tr>
<td>Ancillary Data</td>
<td>SCC, SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
<td></td>
</tr>
<tr>
<td>Teletext</td>
<td>SMI, SMPTE-TT, SRT, TTML, EBU-TT-D, Web-VTT</td>
<td></td>
</tr>
<tr>
<td>DVBSUB</td>
<td>SMPTE-TT</td>
<td></td>
</tr>
<tr>
<td>SCTE-27</td>
<td>SMPTE-TT</td>
<td></td>
</tr>
<tr>
<td>Codec or Container</td>
<td>Direction</td>
<td>Statement</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>444 and 4444 are converted to Apple Prores 422 during input handling.</td>
<td></td>
<td>444 and 4444 are converted to Apple Prores 422 during input handling.</td>
</tr>
<tr>
<td>MPEG-2 video codec</td>
<td>Input</td>
<td>Complete list of supported codecs is: MPEG-2, and ATSC (A/53).</td>
</tr>
<tr>
<td>AAC audio codec</td>
<td>Input</td>
<td>Complete list of supported profiles is: LC-AAC, HE-AAC v1, and HE-AAC v2.</td>
</tr>
<tr>
<td>Dolby Digital audio codec</td>
<td>Input</td>
<td>Dolby Digital is also known as AC-3.</td>
</tr>
<tr>
<td>Dolby Digital audio codec</td>
<td>Input</td>
<td>Dolby Digital is a licensed codec; however, no license is required to decode this codec in input.</td>
</tr>
<tr>
<td>Dolby Digital audio codec</td>
<td>Input</td>
<td>For more notes, see Transcode Support and Passthrough Support for Dolby Audio Codecs (p. 281).</td>
</tr>
<tr>
<td>Dolby Digital Plus audio codec</td>
<td>Input</td>
<td>Dolby Digital Plus is also known as Enhanced AC-3 and is frequently abbreviated as DD+ or EC-3 and E-AC-3. Decoding of Dolby Digital Plus requires the Elemental Audio Decode Package license option.</td>
</tr>
<tr>
<td>Dolby Digital Plus audio codec</td>
<td>Input</td>
<td>For more notes, see Transcode Support and Passthrough Support for Dolby Audio Codecs (p. 281).</td>
</tr>
<tr>
<td>Dolby E frames carried in PCM audio streams tagged with SMPTE-337</td>
<td>Input</td>
<td>Decoding of Dolby E in PCM stream requires the Elemental Audio Decode Package license option.</td>
</tr>
<tr>
<td>Apple ProRes video codec in output</td>
<td>Output</td>
<td>Complete list of supported codecs is: Apple Prores 422 (all profiles).</td>
</tr>
<tr>
<td>Dolby Digital audio codec</td>
<td>Output</td>
<td>Encoding with Dolby Digital requires the Elemental Advanced Audio Package license option.</td>
</tr>
<tr>
<td>Dolby Digital audio codec</td>
<td>Output</td>
<td>For more notes, see Transcode Support and Passthrough Support for Dolby Audio Codecs (p. 281).</td>
</tr>
<tr>
<td>Codec or Container</td>
<td>Direction</td>
<td>Statement</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Dolby Digital Plus audio codec</td>
<td>Output</td>
<td>Encoding with Dolby Digital Plus requires the Elemental Advanced Audio Package license option. For more notes, see Transcode Support and Passthrough Support for Dolby Audio Codecs (p. 281).</td>
</tr>
<tr>
<td>Dolby E pass-through</td>
<td>Output</td>
<td>See Transcode Support and Passthrough Support for Dolby Audio Codecs (p. 281).</td>
</tr>
<tr>
<td>DTS Express</td>
<td>Output</td>
<td>Encoding with DTS Express requires the Elemental Advanced Audio Package license option.</td>
</tr>
<tr>
<td>MPEG Audio codec</td>
<td>Output</td>
<td>MPEG-1 Audio Layer II only</td>
</tr>
</tbody>
</table>

**H.264 (AVC) Support**

H.264 is supported in the following variations.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Chroma Sampling</th>
<th>Bit Depth</th>
<th>Profile/Format</th>
<th>Level</th>
<th>License Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>4:2:0</td>
<td>8-bit and 10-bit</td>
<td>Baseline, Main, High, High 10, High 4:2:2, High 10 Intra, High 422 Intra, AS-11/RP2027</td>
<td>1.0-5.2</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>4:2:2</td>
<td>8-bit and 10-bit</td>
<td>Baseline, Main, High, High 10, High 4:2:2, High 10 Intra, High 422 Intra, AS-11/RP2027</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Output</td>
<td>4:2:0</td>
<td>8-bit</td>
<td>Baseline, Main, High, High 10, High 4:2:2, High 10 Intra, High 422 Intra, AS-11/RP2027</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>4:2:0</td>
<td>10-bit</td>
<td>For AVC Intra, purchase the BCE license pack.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4:2:2</td>
<td>8-bit and 10-bit</td>
<td>For AVC Intra, purchase the BCE license pack.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**HEVC (H.265) Support**

HEVC is supported in the following variations.
**Direction** | **Chroma Sampling** | **Bit Depth** | **Profile/Format** | **Level** | **License Requirement**
---|---|---|---|---|---
**Input** | 4:2:0 | 8-bit and 10-bit | Main, Main 10, Main 4:2:2 10 | 1.0-5.2 | None
 | 4:2:2 | 8-bit and 10-bit | | | None
**Output** | 4:2:0 | 8-bit and 10-bit | | | For HEVC 4:2:0, request the HC license.
 | 4:2:2 | 8-bit and 10-bit | | | For HEVC 4:2:2, purchase the BCE license pack.

**MPEG-2 (H.262) Support**

MPEG-2 is supported in the following variations.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Chroma Sampling</th>
<th>Bit Depth</th>
<th>Profile/Format</th>
<th>Level</th>
<th>License Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>4:2:0, 4:2:2</td>
<td>8-bit</td>
<td>Simple, Main, 422, IMX/D-10</td>
<td>Low, Main, High1440, High</td>
<td>N/A</td>
</tr>
<tr>
<td>Output</td>
<td>4:2:0, 4:2:2</td>
<td>8-bit</td>
<td>Simple, Main, 422, IMX/D-10</td>
<td>Low, Main, High1440, High</td>
<td>N/A</td>
</tr>
</tbody>
</table>

In addition to these general support statements, there are constraints depending on the container. These constraints are described in the section called “Containers and Codecs by Input/Output” (p. 282).

**Codec Support for CPU-Only and GPU-Enabled Deployments**

The codecs listed in the table below are supported with equivalent video quality at equivalent operating points on both CPU-only and GPU-enabled appliances. VC-1 encoding can only be done on GPU-enabled appliances.

<table>
<thead>
<tr>
<th>Output Codec</th>
<th>Equivalent VQ, CPU-Only and GPU-Enabled Systems</th>
<th>Accelerated Encoding on GPU-Enabled Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.264 (AVC)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HEVC (H.265)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MPEG-2 (H.262)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ProRes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Uncompressed YUV</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
AWS Elemental Live User Guide
General Information on Codecs and Containers

Output Codec | Equivalent VQ, CPU-Only and GPU-Enabled Systems | Accelerated Encoding on GPU-Enabled Systems
---|---|---
JPEG (Frame Capture) | Yes | No

Transcode Support and Passthrough Support for Dolby Audio Codecs

Three Dolby codecs are supported and can be transcoded or passed through as follows:

- **Dolby Digital (AC-3):** Can be re-encoded as Dolby Digital or the original Dolby Digital can be passed through.
- **Dolby Digital Plus (E-AC-3):** Can be re-encoded as Dolby Digital Plus or the original Dolby Digital Plus can be passed through.
- **Dolby E:** Can be passed through as Dolby E. Cannot be re-encoded as Dolby E.

The following table specifies the fields on the input side and on the output side that control passthrough versus transcoding.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output &gt; Stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>Codec Detected in Input</td>
<td>Value in Unwrap SMPTE 337 Field</td>
</tr>
<tr>
<td>Dolby Digital</td>
<td>n/a</td>
</tr>
<tr>
<td>Dolby Digital</td>
<td>n/a</td>
</tr>
<tr>
<td>Dolby Digital Plus</td>
<td>n/a</td>
</tr>
<tr>
<td>Dolby Digital Plus</td>
<td>n/a</td>
</tr>
<tr>
<td>Dolby Digital Plus</td>
<td>n/a</td>
</tr>
<tr>
<td>Dolby Digital Plus and a non-Dolby Digital Plus codec</td>
<td>n/a</td>
</tr>
<tr>
<td>Dolby Digital Plus and a non-Dolby Digital Plus codec</td>
<td>n/a</td>
</tr>
</tbody>
</table>
### Input

<table>
<thead>
<tr>
<th>Input</th>
<th>Output &gt; Stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dolby Digital Plus and a non-Dolby Digital Plus codec</td>
<td>n/a</td>
</tr>
<tr>
<td>Dolby E in a PCM stream tagged with SMPTE 337 Unchecked</td>
<td>Uncompressed WAV</td>
</tr>
<tr>
<td>Dolby E in a PCM stream tagged with SMPTE 337 Unchecked</td>
<td>Uncompressed AIFF</td>
</tr>
</tbody>
</table>

### Rules for Ingesting Apple HLS TS Sources

For video, each AWS Elemental Live event can extract only one video from only one rendition. AWS Elemental Live will not reject inputs that contain multiple renditions, but it will handle only one of the renditions. There are fields in the event for specifying which video to extract.

For audio, each AWS Elemental Live event can extract audio from the same rendition as the selected video. It can extract more than one audio from that rendition. It cannot extract audio from two different renditions. There are fields in the event for specifying which audio to extract.

AWS Elemental Live cannot extract audio from a rendition that contains only audio; it does not support ingest of audio rendition groups.

In all cases, the incoming HLS stream must include a manifest.

### Rules for Ingesting MPEG-TS Programs

For video, each AWS Elemental Live event can extract only one video from only one program. AWS Elemental Live will not reject MPTS inputs, but it will handle only one program. There are fields in the event that specify which video to extract.

For audio, each AWS Elemental Live event can extract audio that is in the same program as the video. It can extract more than one audio from that program. It cannot extract audio from another program. There are fields in the event that specify which audio to extract.

### Containers and Codecs by Input/Output

**Topics**

- Containers and Codecs for File Inputs (p. 283)
- Containers and Codecs for Real-Time Inputs (p. 286)
- Containers and Codecs for Archive Outputs (p. 286)
- Containers and Codecs for Real Time Outputs (p. 288)
## Containers and Codecs for File Inputs

<table>
<thead>
<tr>
<th>Container</th>
<th>Media Type</th>
<th>Extensions</th>
<th>Video Codecs</th>
<th>Audio Codecs</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Container</td>
<td></td>
<td>.m2v, .m1v</td>
<td>DV/DVCPRO, H.264, HEVC (H.265), MPEG-1, MPEG-2</td>
<td></td>
</tr>
<tr>
<td>Apple HLS</td>
<td>HLS</td>
<td>.m3u8</td>
<td>H.264, HEVC (H.265)</td>
<td>AAC</td>
</tr>
<tr>
<td>Audio Video Interleave</td>
<td>AVI</td>
<td>.avi, .divx, .xvid</td>
<td>Uncompressed, DivX/Xvid, DV/DVCPRO</td>
<td>Dolby Digital, Dolby Digital Plus up to 7.1, Dolby E frames carried in PCM streams tagged with SMPTE-337, MPEG Audio, PCM</td>
</tr>
<tr>
<td>Adobe Flash</td>
<td>F4V</td>
<td>.f4v, .flv</td>
<td>H.263, H.264</td>
<td>AAC</td>
</tr>
<tr>
<td>Matroska</td>
<td>MKV</td>
<td>.mkv</td>
<td>H.264, MPEG-2, MPEG-4 part 2, VC-1</td>
<td>AAC, Dolby Digital, Dolby Digital Plus up to 7.1, WMA, WMA2</td>
</tr>
<tr>
<td>MPEG Transport Streams</td>
<td>MPEG TS</td>
<td>.m2ts, .m2t, .mts, .ts</td>
<td>H.264, HEVC (H.265), MPEG-2, VC-1</td>
<td>AAC, AIFF, Dolby Digital, Dolby Digital Plus up to 7.1, Dolby E frames carried in non-PCM streams. Supported with file inputs, not with IP inputs.</td>
</tr>
<tr>
<td>Container</td>
<td>Media Type</td>
<td>Extensions</td>
<td>Video Codecs</td>
<td>Audio Codecs</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>------------</td>
<td>--------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>MPEG-1 System Streams</td>
<td>MPEG SS</td>
<td>.mpg, .mpeg</td>
<td>MPEG-1, MPEG-2</td>
<td>AAC, AIFF, Dolby Digital, Dolby Digital Plus up to 7.1, MPEG Audio, PCM</td>
</tr>
<tr>
<td>MPEG-4</td>
<td>MPEG-4</td>
<td>.mp4, .m4v, .f4v</td>
<td>Uncompressed, AVC Intra 50/100, DivX/Xvid, H.261, H.262, H.263, H.264, JPEG 2000, MJPEG, MPEG-2, MPEG-4 part 2, VC-1</td>
<td>AAC, Dolby Digital, Dolby Digital Plus up to 7.1, PCM, WMA, WMA2</td>
</tr>
<tr>
<td>Container</td>
<td>Media Type</td>
<td>Extensions</td>
<td>Video Codecs</td>
<td>Audio Codecs</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>-------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>MXF</td>
<td>MXF</td>
<td>.mxf</td>
<td>Uncompressed, AVC Intra 50/100, DnxHD, DV/DVCPRO, DV25, DV50, DVCPRO HD, H.264, JPEG 2000, MPEG-2</td>
<td>AAC, AIFF, MPEG Audio, PCM, Dolby E frames carried in PCM streams tagged with SMPTE-337</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QuickTime</td>
<td></td>
<td>.mov</td>
<td>Uncompressed, Apple ProRes, AVC Intra 50/100, DivX/Xvid, DV/DVCPRO, H.261, H.262, H.263, H.264, JPEG 2000, MJPEG, MPEG-2, MPEG-4 part 2</td>
<td>AAC, Dolby E frames carried in PCM streams tagged with SMPTE-337</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAV</td>
<td>WAV</td>
<td>.wav</td>
<td></td>
<td>Dolby E frames carried in PCM streams tagged with SMPTE-337</td>
</tr>
<tr>
<td>WMV/ASF</td>
<td>WMV/ASF</td>
<td>.wmv, .asf</td>
<td>VC-1</td>
<td>WMA, WMA2</td>
</tr>
</tbody>
</table>
Containers and Codecs for Real-Time Inputs

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Video Codecs</th>
<th>Audio Codecs</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDI</td>
<td>Uncompressed</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby E frames carried in PCM streams tagged with SMPTE-337</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCM</td>
</tr>
<tr>
<td>UDP/RTP</td>
<td>MPEG TS</td>
<td>H.264</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HEVC (H.265)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPEG-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPEG Audio</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCM</td>
</tr>
<tr>
<td>ASI</td>
<td>MPEG TS</td>
<td>H.264</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HEVC (H.265)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPEG-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPEG Audio</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCM</td>
</tr>
<tr>
<td>HTTP</td>
<td>HLS</td>
<td>H.264</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HEVC (H.265)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AAC</td>
</tr>
<tr>
<td>RTMP</td>
<td>H.264</td>
<td>AAC</td>
</tr>
</tbody>
</table>

Containers and Codecs for Archive Outputs

<table>
<thead>
<tr>
<th>Container</th>
<th>Video Codecs</th>
<th>Audio Codecs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw (No container)</td>
<td>Frame Capture (MJPEG)</td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td>H.264*</td>
<td>AIFF</td>
</tr>
<tr>
<td></td>
<td>HEVC (H.265)**</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td>MJPEG</td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td></td>
<td>MPEG2</td>
<td>DTS Express</td>
</tr>
<tr>
<td></td>
<td>Uncompressed (see below)</td>
<td>MPEG Audio</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAV</td>
</tr>
<tr>
<td>Container</td>
<td>Video Codecs</td>
<td>Audio Codecs</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Apple HLS</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td>HEVC (H.265)**</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td>3GPP</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td>MXF</td>
<td>MPEG-2</td>
<td>WAV</td>
</tr>
<tr>
<td>ISMV for MSS</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td>HEVC (H.264)</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td>VC-1</td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WMA2</td>
</tr>
<tr>
<td>MPEG DASH ISO</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td>HEVC (H.265)**</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td>MPEG-2 Transport Stream</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td>HEVC (H.265)**</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td>MPEG2</td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPEG Audio</td>
</tr>
<tr>
<td>MPEG-4 (.mp4)</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td>HEVC (H.265)**</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DTS Express</td>
</tr>
<tr>
<td>MPEG-4 Flash (.f4v)</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td>QuickTime</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td>MPEG2</td>
<td>AIFF</td>
</tr>
<tr>
<td></td>
<td>Apple ProRes</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td>Uncompressed (see below)</td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAV</td>
</tr>
<tr>
<td>WMV/ASF</td>
<td>VC-1</td>
<td>WMA2</td>
</tr>
<tr>
<td>XDCAM</td>
<td>MPEG2</td>
<td>WAV</td>
</tr>
<tr>
<td>YUV4MPEG2</td>
<td>Uncompressed</td>
<td>N/A***</td>
</tr>
</tbody>
</table>
*Both 4:2:0 and 4:2:2 H.264 are supported in raw outputs (.264, .avc, extensions) and MPEG-2 transport streams. In all other containers, only 4:2:0 H.264 is supported.

** Both 4:2:0 and 4:2:2 HEVC are supported in raw outputs (.264, .avc, extensions) and MPEG-2 transport streams. In all other containers, only 4:2:0 HEVC is supported.

*** This container does not support audio.

**Uncompressed Formats that Are Supported in Raw Outputs**

<table>
<thead>
<tr>
<th>Pixel Formats</th>
<th>4CC Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit 4:2:0 planar</td>
<td>YV12</td>
</tr>
<tr>
<td>8-bit 4:2:0 planar</td>
<td>I420, IYUV</td>
</tr>
<tr>
<td>8-bit 4:2:0 chroma interleaved</td>
<td>NV12</td>
</tr>
<tr>
<td>8-bit 4:2:2 planar</td>
<td>YV16, I422</td>
</tr>
<tr>
<td>8-bit 4:2:2 planar</td>
<td>UYVY, Y422, UYNV, 2vuy</td>
</tr>
<tr>
<td>8-bit 4:2:2 planar</td>
<td>YUYV, YUY2, YUNV, V422, yuvs</td>
</tr>
<tr>
<td>10-bit 4:2:2 packet</td>
<td>v210, P210</td>
</tr>
</tbody>
</table>

**Containers and Codecs for Real Time Outputs**

<table>
<thead>
<tr>
<th>Container</th>
<th>Video Codecs</th>
<th>Audio Codecs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple HLS</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td>HEVC (H.265)**</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td>DASH-ISO</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td>HEVC (H.265)**</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td>HLS fMP4</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td>HEVC (H.265)**</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td>ISMV for MSS</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td></td>
<td>HEVC (H.265)</td>
<td>Dolby Digital</td>
</tr>
<tr>
<td></td>
<td>VC-1</td>
<td>Dolby Digital Plus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WMA2</td>
</tr>
<tr>
<td>RTMP</td>
<td>H.264*</td>
<td>AAC</td>
</tr>
<tr>
<td>UDP/TS</td>
<td>H.264*</td>
<td>Dolby Digital</td>
</tr>
</tbody>
</table>
Reference: Supported DRM Solutions

This chapter lists the Digital Rights Management (DRM) solutions that are available for the types of outputs supported by AWS Elemental Live. The tables are organized alphabetically by output type and the DRM technology provider name.

Topics
- DASH Output (p. 289)
- HLS Output with Apple FairPlay (p. 290)
- HLS Output with PlayReady (p. 291)
- HLS Output with SecureMedia (p. 293)
- HLS Output with Verimatrix (p. 293)
- Microsoft Smooth Output with PlayReady (p. 294)
- UDP/TS Outputs with DVB Simulcrypt Standard (p. 295)

### DASH Output

Encryption mode: Always AES CTR (AES-128)

Key rotation: Always Static

Support client players: Consult with the DRM solution provider (for SPEKE) or the DRM Technology provider for supported players.

<table>
<thead>
<tr>
<th>Description</th>
<th>DRM Technology Provider</th>
<th>Key Provider (DRM Implementer)</th>
<th>Version of Server API from DRM Implementer</th>
</tr>
</thead>
<tbody>
<tr>
<td>The customer uses a SPEKE-compliant DRM solution for protecting DASH output using Widevine/CENC and PlayReady/CENC technology.</td>
<td>CENC&lt;br&gt;Widevine and PlayReady</td>
<td>SPEKE</td>
<td>SPEKE v1.0</td>
</tr>
<tr>
<td>The customer uses a static PlayReady key for protecting DASH output using PlayReady/CENC technology.</td>
<td>PlayReady</td>
<td>PlayReady</td>
<td>Not applicable</td>
</tr>
<tr>
<td>The customer uses a static Widevine key for protecting DASH output using Widevine/CENC technology.</td>
<td>CENC/Widevine</td>
<td>Generic</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>
### HLS Output with Apple FairPlay

**Encryption mode:** Always AES CBC (Sample AES)

**Supported client players:** Consult with the DRM solution provider (for SPEKE) or the key provider (DRM implementer) for supported players. For a self-generated key, the iOS players must be able to retrieve the key. For a generic key, the iOS player must be able to retrieve the key from the manifest.

<table>
<thead>
<tr>
<th>Description</th>
<th>Key Provider (DRM Implementer)</th>
<th>Version of Server API from DRM Implementer</th>
<th>Key Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The customer uses a SPEKE-compliant DRM solution for protecting HLS fMP4 output using Apple Fairplay DRM technology.</td>
<td>SPEKE</td>
<td>SPEKE v1.0</td>
<td>Static, Rotating</td>
</tr>
<tr>
<td>The customer uses a SPEKE-compliant DRM solution for protecting HLS output using Apple Fairplay DRM technology.</td>
<td>SPEKE</td>
<td>SPEKE v1.0</td>
<td>Static, Rotating</td>
</tr>
<tr>
<td>The customer uses the 1Mainstream DRM solution for protecting HLS output using the Apple Fairplay DRM technology. The end user plays the content on a 1Mainstream-approved player.</td>
<td>1Mainstream</td>
<td>Version 1.1</td>
<td>Static</td>
</tr>
<tr>
<td>AWS Elemental Live generates a key that it uses to encrypt the content. AWS Elemental Live also puts that key</td>
<td>Self-Generated</td>
<td>Not applicable; key generated by AWS Elemental Live</td>
<td>Static, Rotating</td>
</tr>
</tbody>
</table>
### HLS Output with PlayReady

**Encryption mode:** Always AES CTR (AES-128)

**Supported client players:** Consult with the key provider (DRM implementer) for supported players.

<table>
<thead>
<tr>
<th>Description</th>
<th>Key Provider (DRM Implementer)</th>
<th>Version of Server API from DRM Implementer</th>
<th>Key Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The customer uses the Conax DRM solution for protecting HLS output using the PlayReady DRM technology. The end user plays the content on a Conax-approved player.</td>
<td>Conax</td>
<td>GetPlayReady KeyForHLS v1</td>
<td>Static</td>
</tr>
<tr>
<td>The customer uses the Irdeto DRM. (ActiveCloak for Media) solution for protecting HLS output using the PlayReady DRM</td>
<td>Irdeto ActiveCloak for Media</td>
<td>Irdeto does not currently have API versioning.</td>
<td>Static</td>
</tr>
<tr>
<td>Description</td>
<td>Key Provider (DRM Implementer)</td>
<td>Version of Server API from DRM Implementer</td>
<td>Key Rotation</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------</td>
<td>---------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>The customer uses the InsideSecure DRM solution for protecting HLS output using the PlayReady DRM technology. The end user plays the content on an Inside-Secure-approved player.</td>
<td>InsideSecure</td>
<td>keyprovisioning v1.0</td>
<td>Static, Rotating</td>
</tr>
<tr>
<td>The customer uses the InsideSecure feature of the thePlatform DRM solution for protecting HLS output using the PlayReady DRM technology. The end user plays the content on a thePlatform-approved player.</td>
<td>InsideSecure on thePlatform</td>
<td>Not applicable; static key generated by AWS Elemental Live</td>
<td>Static</td>
</tr>
<tr>
<td>The customer uses the Irdeto feature of the thePlatform DRM solution for protecting HLS output using the PlayReady DRM technology. The end user will play the content on a thePlatform-approved player.</td>
<td>Irdeto on thePlatform</td>
<td>Not applicable; static key generated by AWS Elemental Live</td>
<td>Static</td>
</tr>
<tr>
<td>The customer uses the Microsoft feature of the thePlatform DRM solution for protecting HLS output using the PlayReady DRM technology. The end user plays the content on a thePlatform-approved player.</td>
<td>Microsoft client on thePlatform</td>
<td>Not applicable; static key generated by AWS Elemental Live</td>
<td>Static</td>
</tr>
<tr>
<td>Description</td>
<td>Key Provider (DRM Implementer)</td>
<td>Version of Server API from DRM Implementer</td>
<td>Key Rotation</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------</td>
<td>-------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>The customer uses the Piksel DRM solution for protecting HLS output using the PlayReady DRM technology. The end user will play the content on a Piksel-approved player.</td>
<td>Piksel</td>
<td>GetEncryptInfo v1.0</td>
<td>Static</td>
</tr>
</tbody>
</table>

**HLS Output with SecureMedia**

Encryption mode: Always AES CTR (AES-128)

Supported client players: Consult with the key provider (DRM implementer) for supported players.

<table>
<thead>
<tr>
<th>Description</th>
<th>Key Provider (DRM Implementer)</th>
<th>Version of Server API from DRM Implementer</th>
<th>Key Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The customer uses the Arris SecureMedia DRM solution for protecting HLS output using the SecureMedia DRM technology. The end user plays the content on a SecureMedia-approved player.</td>
<td>SecureMedia</td>
<td>No versioning information is available from Arris.</td>
<td>Static, Rotating</td>
</tr>
</tbody>
</table>

**HLS Output with Verimatrix**

Encryption mode: Always AES CTR (AES-128)

Supported client players: Consult with the key provider (DRM implementer) for supported players.

<table>
<thead>
<tr>
<th>Description</th>
<th>DRM Technology Provider</th>
<th>Key Provider (DRM Implementer)</th>
<th>Version of Server API from DRM Implementer</th>
<th>Client Player</th>
<th>Encryption Mode</th>
<th>Key Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The customer uses the Verimatrix VCAS DRM solution for protecting HLS output. This solution</td>
<td>Verimatrix Content Authority System (VCAS)</td>
<td>Verimatrix</td>
<td>VCAS for Internet TV 4.2 Integration Guide</td>
<td>Verimatrix-approved player</td>
<td>AES CBC (AES-128)</td>
<td>Static, Rotating</td>
</tr>
<tr>
<td>Description</td>
<td>DRM Technology Provider</td>
<td>Key Provider (DRM Implementer)</td>
<td>Version of Server API from DRM Implementer</td>
<td>Client Player</td>
<td>Encryption Mode</td>
<td>Key Rotation</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------</td>
<td>---------------------------------</td>
<td>------------------------------------------</td>
<td>---------------</td>
<td>----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>uses the Verimatrix-proprietary DRM technology. The end user plays the content on a Verimatrix-approved player.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Microsoft Smooth Output with PlayReady**

Encryption mode: Always AES CTR (AES-128)

Supported client players: Consult with the key provider (DRM implementer) for supported players.

<table>
<thead>
<tr>
<th>Description</th>
<th>Key Provider (DRM Implementer)</th>
<th>Version of Server API from DRM Implementer</th>
<th>Key Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The customer uses the Conax DRM solution for protecting MSS output using the PlayReady DRM technology. The end user plays the content on a Conax-approved player.</td>
<td>Conax</td>
<td>GetPlayReady KeyFor MSSmooth Streaming v1</td>
<td>Static</td>
</tr>
<tr>
<td>The customer uses the InsideSecure DRM solution for protecting MSS output using the PlayReady DRM technology. The end user plays the content on an InsideSecure-approved player.</td>
<td>InsideSecure</td>
<td>keyprovisioning v1.0</td>
<td>Rotating</td>
</tr>
<tr>
<td>The customer uses the Irdeto feature of the thePlatform DRM solution for protecting MSS output using the PlayReady DRM technology. The end user plays the content on a thePlatform-approved player.</td>
<td>Irdeto on the Platform</td>
<td>Not applicable; static key generated by AWS Elemental Live.</td>
<td>Static</td>
</tr>
</tbody>
</table>
### UDP/TS Outputs with DVB Simulcrypt Standard

**Description**

- The customer uses the Piksel DRM solution for protecting MSS output using the PlayReady DRM technology. The end user plays the content on a Piksel-approved player.

  **Key Provider (DRM Implementer):** Piksel  
  **Version of Server API from DRM Implementer:** GetEncryptInfo v1.0  
  **Key Rotation:** Static

- AWS Elemental Live lets you enter a key or generate a key that AWS Elemental Live uses to encrypt the content. AWS Elemental Live also puts that key at a customer-specified location; the client player retrieves the key from that location and decrypts the content. The end user plays the content on a Microsoft-Silverlight-approved player.

  Strictly speaking, an encryption solution, not a DRM solution.

  **Key Provider (DRM Implementer):** Self-Generated or Static  
  **Version of Server API from DRM Implementer:** Not applicable.  
  **Key Rotation:** Static

- The customer uses the Seachange DRM solution for protecting MSS output using the PlayReady DRM technology. The end user plays the content on a Seachange-approved player.

  **Key Provider (DRM Implementer):** Seachange  
  **Version of Server API from DRM Implementer:** Acquire Packaging Data v1.0  
  **Key Rotation:** Static

---

**UDP/TS Outputs with DVB Simulcrypt Standard**

Encryption mode: Always AES CBC as described in ATIS-0800006

Supported client players: Consult with the key provider (DRM implementer) for supported players.
### Description

<table>
<thead>
<tr>
<th>Description</th>
<th>Key Provider (DRM Implementer)</th>
<th>Version of Server API from DRM Implementer</th>
<th>Key Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The customer uses the Verimatrix MultiCAS/DVB DRM solution for protecting UDP/TS output in compliance with the DVB Simulcrypt standard. The end user plays the content on a Verimatrix-approved player.</td>
<td>Verimatrix</td>
<td>ECMG interface as described in ETSI TS 101 197</td>
<td>Rotating</td>
</tr>
</tbody>
</table>
Document History for User Guide

The following table describes the documentation for this release of AWS Elemental Live.

- **API version:** 2.17
- **Release notes:** [AWS Elemental Live Release Notes](#)

The following table describes the documentation for this release of AWS Elemental Live. For notification about updates to this documentation, you can subscribe to an RSS feed.

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 2.17 release (p. 1)</td>
<td>Changes to support the 2.17 software release.</td>
<td>January 7, 2020</td>
</tr>
</tbody>
</table>