# AWS Elemental MediaTailor

## User Guide

# AWS Elemental MediaTailor: User Guide



The AWS Documentation website is getting a new look!

Try it now and let us know what you think. Switch to the new look >>

You can return to the original look by selecting English in the language selector above.

# Table of Contents

# What Is AWS Elemental MediaTailor?

AWS Elemental MediaTailor is a scalable ad insertion service that runs in the AWS Cloud. With MediaTailor, you can serve targeted ads to viewers while maintaining broadcast quality in over-the-top (OTT) video applications.

AWS Elemental MediaTailor offers important advances over traditional ad-tracking systems: ads are better monetized, more consistent in video quality and resolution, and easier to manage across multi-platform environments. MediaTailor simplifies your ad workflow by allowing all IP-connected devices to render ads in the same way as they render other content. The service also offers advanced tracking of ad views, which further increases the monetization of content.

MediaTailor supports Apple HTTP Live Streaming (HLS) and MPEG Dynamic Adaptive Streaming over HTTP (DASH) for video on demand (VOD) and live workflows.

## Concepts and Terminology

Ad decision server (ADS)

A server that provides advertising spot specifications based on criteria including current advertising campaigns and viewer preferences.

Configuration

An object in MediaTailor that you interact with. The configuration holds location information about the origin server and the ad decision server (ADS). The configuration also holds endpoints that provide access points in and out of MediaTailor.

Dynamic transcoding

A process that matches the ad quality and format to the primary video content when content is requested. Dynamic transcoding reduces storage requirements and ensures that playback seamlessly transitions between the ad and video content.

Manifest manipulation

The process of rewriting manifests from the origin server so that the manifests reference the appropriate ad and content fragments. Ads are determined by the VAST response from the ad decision server (ADS). As playback progresses, MediaTailor performs ad insertion or ad replacement into the content stream.

VAST and VMAP

Video Ad Serving Template (VAST) and Video Multiple Ad Playlist (VMAP) are XML responses that the ad decision server (ADS) sends to ad requests from MediaTailor. The responses dictate what ads MediaTailor inserts in the manifest. VMAP also includes timing for ad avails. For more information about the logic behind MediaTailor ad insertion, see Ad Behavior in AWS Elemental MediaTailor (p. 66). For more information about how MediaTailor works with VAST, see *VAST* (p. 57).

## How MediaTailor Works

MediaTailor serves personalized content to viewers while maintaining broadcast quality-of-service in over-the-top (OTT) applications.

Here is the general MediaTailor processing flow:

1. A player or content distribution network (CDN) such as Amazon CloudFront sends a request to MediaTailor for HLS or DASH content. The request contains parameters from the player that includes information about the viewer that is used for ad customization. The format of the request varies depending on whether you use server-side or client-side reporting to track how much of an ad the viewer watches.

   For information about how the requests differ between the two reporting methods, see Ad Tracking Reporting in AWS Elemental MediaTailor (p. 70). For information about configuring the ad targeting parameters, see Dynamic Ad Variables in AWS Elemental MediaTailor (p. 60).
2. Based on the request, MediaTailor retrieves the content manifest and ad specifications as follows:
   - MediaTailor sends a manifest request to its origin server, typically AWS Elemental MediaPackage. The origin server returns a fully formed template manifest with ad markers, so that MediaTailor knows where to perform ad insertion or replacement.
   - MediaTailor sends a request that includes the viewer information to its ADS. The ADS chooses ads based on the viewer information and current ad campaigns. It returns the ad URLs in a VAST or VMAP response.
3. MediaTailor manipulates the manifest to include the URLs for the appropriate ads from the VAST or VMAP response. For the logic behind how ads are inserted, see Ad Behavior in AWS Elemental MediaTailor (p. 66).
4. MediaTailor provides the fully customized manifest to the requesting CDN or player.
5. As playback progresses, either MediaTailor or the video player reports how much of an ad is played to the ADS ad tracking URL. For more information about ad reporting, see *Ad Tracking Reporting* (p. 70). The player requests ad segments throughout content playback. When MediaTailor receives an ad segment request, if the ad is not already transcoded in a format that matches the video content, MediaTailor transcodes the ad. If an ad is not already transcoded, MediaTailor doesn't present it for playback at the first request.

# Mixed Content Requests

Content requests are mixed when some requests are sent over HTTPS, while others are sent over HTTP. Player requests for manifests and ad segments from MediaTailor are always sent over HTTPS. If the origin server accepts only HTTP requests, playback might fail at the player. To avoid playback issues, do one of the following:

- Use an origin server that supports HTTPS requests.
- Use a content distribution network (CDN) to enforce HTTPS requests. For more information, see Using HTTPS in Amazon CloudFront.

# Manifest Response Latency

A certain amount of latency is normal for MediaTailor responses to manifests. Latency occurs mainly for these three reasons:

- Manifest processing latency – time it takes for MediaTailor to look up entries in databases, and to compute and produce manifests. Latency is usually less than 100 milliseconds.
- ADS latency – time it takes for the ADS to respond to the MediaTailor request. Latency is variable, but MediaTailor times out if the ADS hasn't sent a response in 1.5 seconds or less.
- Origin server latency – time it takes for the origin server to respond to the MediaTailor request. Latency is variable, but MediaTailor times out if the origin server hasn't sent a response in 2 seconds or less.

# Features of MediaTailor

MediaTailor supports the following features:

Ad Tracking Reporting

MediaTailor offers server-side ad view reporting for HLS and client-side ad view reporting for HLS and DASH:

- For server-side reporting, the service sends reporting information to ad tracking URLs directly.
- For client-side reporting, the service provides the beacons for the downstream player or content distribution network (CDN) to use. The player or CDN directly calls the ADS to report how much of an ad that a viewer watches, in quartile percentages (25%, 50%, 75%, or 100%).

For more information about setting up reporting, see Ad Tracking Reporting in AWS Elemental MediaTailor (p. 70).

Audio

MediaTailor supports multiple audio tracks. For more information, see Alternate Audio and Subtitles (p. 17).

Content and Ad Continuity

MediaTailor uses a transcoding service to ensure that ads and content have the same bitrate and resolution so that transitions are smooth throughout playback.

MediaTailor also performs audio normalization between ads and main content. This normalization ensures that the playback volume stays consistent between ads and the content.

Personalized Content

MediaTailor uses VAST or VMAP to pass viewer information to the ad decision server (ADS), and in return receives targeted ads that are relevant for the viewer.

# Related Services

- **Amazon CloudFront** is a global content delivery network (CDN) service that securely delivers data and videos to your viewers. Use CloudFront to deliver content with the best possible performance. For more information about CloudFront, see the Amazon CloudFront website.
- **AWS Elemental MediaPackage** is a just-in-time packaging and origination service that customizes live video assets for distribution in a format that is compatible with the device that makes the request. Use AWS Elemental MediaPackage as an origin server to prepare content and add ad markers before sending streams to MediaTailor. For more information about how MediaTailor works with origin servers, see How MediaTailor Works (p. 1).
- **AWS Identity and Access Management (IAM)** is a web service that helps you securely control access to AWS resources for your users. Use IAM to control who can use your AWS resources (authentication) and what resources they can use in which ways (authorization). For more information, see Setting Up AWS Elemental MediaTailor (p. 5).

# Accessing MediaTailor

You can access MediaTailor using the service's console.

Access your AWS account by providing credentials that verify that you have permissions to use the services.

To log in to the MediaTailor console, use the following link: **`https://console.aws.amazon.com/mediatailor/home`**.

# Pricing for MediaTailor

As with other AWS products, there are no contracts or minimum commitments for using MediaTailor. You are charged based on your use of the service. For more information, see MediaTailor Pricing.

# Regions for MediaTailor

To reduce data latency in your applications, MediaTailor offers regional endpoints to make your requests. To view the list of Regions in which MediaTailor is available, see https://docs.aws.amazon.com/general/latest/gr/rande.html#mediatailor_region.

# Stream Requirements

A video stream must meet the following requirements to work with MediaTailor:

- Use Apple HLS (HTTP Live Streaming) or MPEG DASH (Dynamic Adaptive Streaming over HTTP)
- Use live streaming or video on demand (VOD)
- Be accessible on the public internet and have a public IP address
- Contain ad markers in one of the formats described in Step 2: Prepare a Stream (p. 10)

# Transcoded Ad Management

MediaTailor manages transcoded ads on your behalf with no additional charge. When you play an ad in a video stream, it might get copied to another AWS Region.

If you need to delete your transcoded ad assets for any reason, file a case with AWS Support. On the navigation bar of the console, choose **Support**, and then choose **Support Center**. Create a case, and choose the category of **Service Limit Increase**.

# Setting Up AWS Elemental MediaTailor

This section guides you through the steps required to configure users to access AWS Elemental MediaTailor. For background and additional information about identity and access management for MediaTailor, see Identity and Access Management for AWS Elemental MediaTailor (p. 83).

To start using AWS Elemental MediaTailor, complete the following steps.

**Topics**



## Signing Up for AWS

If you do not have an AWS account, complete the following steps to create one.

**To sign up for an AWS account**

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

## Creating an Admin IAM User

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

In the following procedure, you use the AWS account root user to create your first IAM user. You then add this IAM user to an Administrators group, to ensure that you have access to all services and their resources in your account. The next time that you access your AWS account, sign in with the credentials for this IAM user.

To create users with limited permissions, see Creating a Non-Admin IAM User (p. 6).

**To create an administrator user for yourself and add the user to an administrators group (console)**

1. Use your AWS account email address and password to sign in as the *AWS account root user* to the IAM console at https://console.aws.amazon.com/iam/.

**Note**
We strongly recommend that you adhere to the best practice of using the **`Administrator`** IAM user below and securely lock away the root user credentials. Sign in as the root user only to perform a few account and service management tasks.

2. In the navigation pane, choose **Users** and then choose **Add user**.
3. For **User name**, enter **`Administrator`**.
4. Select the check box next to **AWS Management Console access**. Then select **Custom password**, and then enter your new password in the text box.
5. (Optional) By default, AWS requires the new user to create a new password when first signing in. You can clear the check box next to **User must create a new password at next sign-in** to allow the new user to reset their password after they sign in.
6. Choose **Next: Permissions**.
7. Under **Set permissions**, choose **Add user to group**.
8. Choose **Create group**.
9. In the **Create group** dialog box, for **Group name** enter **`Administrators`**.
10. Choose **Filter policies**, and then select **AWS managed -job function** to filter the table contents.
11. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.

    **Note**
    You must activate IAM user and role access to Billing before you can use the `AdministratorAccess` permissions to access the AWS Billing and Cost Management console. To do this, follow the instructions in step 1 of the tutorial about delegating access to the billing console.
12. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
13. Choose **Next: Tags**.
14. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see Tagging IAM Entities in the *IAM User Guide*.
15. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users and to give your users access to your AWS account resources. To learn about using policies that restrict user permissions to specific AWS resources, see Access Management and Example Policies.

For information about creating users with limited permissions, see Creating a Non-Admin IAM User (p. 6).

# Creating a Non-Admin IAM User

Users in the Administrators group for an account have access to all AWS services and resources in that account. This section describes how to create users with permissions that are limited to AWS Elemental MediaTailor.

**Topics**

# Step 1: Sign In To the IAM Console

Sign in to the IAM console to manage policies and users.

**To sign in**

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

   Use your Administrator user credentials to sign in.

# Step 2: Create Policies

Create two policies for AWS Elemental MediaTailor: one to provide read/write access, and one to provide read-only access.

**To create policies for AWS Elemental MediaTailor**

1. In the navigation pane of the IAM console, choose **Policies**.
2. On the **Policies** page, create a policy named **MediaTailorAllAccess** that allows all actions on all resources in MediaTailor:
   a. Choose **Create policy**.
   b. Choose the **JSON** tab and paste the following policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "mediatailor:*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:DescribeAvailabilityZones"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
         {
            "Action": [
                "cloudwatch:GetMetricStatistics"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

   c. Choose **Review policy**.

d. On the **Review policy** page, for **Name**, enter **`MediaTailorAllAccess`**, and then choose **Create policy**.

3. On the **Policies** page, create a read-only policy named **MediaTailorReadOnlyAccess** for MediaTailor:
   a. Choose **Create policy**.
   b. Choose the **JSON** tab and paste the following read-only policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "mediatailor:GetPlaybackConfiguration",
                "mediatailor:ListPlaybackConfigurations",
                "mediatailor:ListTagsForResource"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:DescribeAvailabilityZones"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
         {
            "Action": [
                "cloudwatch:GetMetricStatistics"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

   c. Choose **Review policy**.
   d. On the **Review policy** page, for **Name**, enter **`MediaTailorReadOnlyAccess`**, and then choose **Create policy**.

# Step 3: Create User Groups

Create a user group for each of the policies that you created in the section called “Step 2: Create Policies” (p. 7). This way, when you create additional users you can add the users to a group rather than attaching individual policies to each user.

**To create groups for users who need access to AWS Elemental MediaTailor**

1. In the navigation pane of the IAM console, choose **Groups**.
2. On the **Groups** page, choose **Create New Group** and create an administrator group using the **MediaTailorAllAccess** policy:
   a. On the **Set Group Name** page, enter a name for the group, such as **`MediaTailorAdmins`**. Choose **Next Step**.

   b. On the **Attach Policy** page, for **Filter**, choose **Customer Managed**.
   c. In the policy list, choose the **MediaTailorAllAccess** policy that you created.
   d. On the **Review** page, verify that the correct policy is added to this group, and then choose **Create Group**.
3. On the **Groups** page, choose **Create New Group** and create a read-only group using the **MediaTailorReadOnlyAccess** policy:
   a. On the **Set Group Name** page, enter a name for the group, such as **`MediaTailorReadOnly`**. Choose **Next Step**.
   b. On the **Attach Policy** page, for **Filter**, choose **Customer Managed**.
   c. In the policy list, choose the **MediaTailorReadOnlyAccess** policy that you created.
   d. On the **Review** page, verify that the correct policy is added to this group, and then choose **Create Group**.

# Step 4: Create Users

Create IAM users for the individuals who require access to AWS Elemental MediaTailor. Next, add each user to the appropriate user group to ensure that they have the right level of permissions. If you already have users created, skip past the user creation steps to modify the permissions for the users.

**To create users who can access AWS Elemental MediaTailor**

1. In the navigation pane of the IAM console, choose **Users**, and then choose **Add user**.
2. For **User name**, enter the name that the user will use to sign in to MediaTailor.
3. Select the check box next to **AWS Management Console access**, select **Custom password**, and then enter the new user's password in the box. You can optionally select **Require password reset** to force the user to create a password the next time the user signs in.
4. Choose **Next: Permissions**.
5. On the **Set permissions for user** page, choose **Add user to group**.
6. Modify the permissions for the users in the group list. Choose the group with the appropriate attached policy. Remember that permissions levels are as follows:
   - The group with the **MediaTailorAllAccess** policy allows all actions on all resources in MediaTailor.
   - The group with the **MediaTailorReadOnlyAccess** policy allows read-only rights for all resources in MediaTailor.
7. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

# Getting Started with AWS Elemental MediaTailor

This Getting Started tutorial shows you how to integrate AWS Elemental MediaTailor into your workflow, including how to create a MediaTailor configuration that holds information about the origin server and ad decision server (ADS).

**Topics**



## Prerequisites

To use AWS Elemental MediaTailor, you need an AWS account and permissions to access, view, and edit MediaTailor configurations. For information on how to do this, see Setting Up AWS Elemental MediaTailor (p. 5).

## Step 1: Access AWS Elemental MediaTailor

Using your IAM credentials, sign in to the MediaTailor console at **`https://console.aws.amazon.com/mediatailor/home`**.

## Step 2: Prepare a Stream

Configure your origin server to produce manifests for HLS or DASH that are compatible with AWS Elemental MediaTailor.

### Prepare an HLS Stream

HLS manifests must satisfy the following requirements:

- Manifests must be accessible on the public internet.
- Manifests must be live or video-on-demand (VOD).

- Manifests must have an `EXT-X-VERSION` of `3` or higher.
- For live content, manifests must contain markers to delineate ad avails. This is optional for VOD content, which can use VMAP timeoffsets instead.

  The manifest file must have ad slots marked with one of the following:

  - **#EXT-X-CUE-OUT / #EXT-X-CUE-IN** (more common) with durations as shown in the following example.

    ```
    #EXT-X-CUE-OUT:60.00
    #EXT-X-CUE-IN
    ```

  - **#EXT-X-DATERANGE** (less common) with durations as shown in the following example.

    ```
    #EXT-X-DATERANGE:ID="",START-DATE="",DURATION=30.000,SCTE35-OUT=0xF
    #EXT-X-DATERANGE:ID="",START-DATE="",DURATION=30.000,SCTE35-OUT=0xF
    ```

    All fields shown for `#EXT-X-DATERANGE` are required.

  The way that you configure the ad markers in the manifest influences whether ads are inserted in a stream or replace other fragments in the stream. For more information, see *Ad Behavior* (p. 66).
- HLS master manifests must follow the HLS specification documented at HTTP Live Streaming: Master Playlist Tags. In particular, `#EXT-X-STREAM-INF` must include the fields `RESOLUTION`, `BANDWIDTH`, and `CODEC`.

After you have configured the stream, note the content origin URL prefix for the master manifest. You need it to create the configuration in AWS Elemental MediaTailor, later in this tutorial.

# Prepare a DASH Stream

DASH manifests must satisfy the following requirements:

- Manifests must be accessible on the public internet.
- Manifests must be live or video on demand (VOD).
- Manifests must mark events as ad avails using either splice insert markers or time signal markers. You can provide the ad markers in clear XML or in base64-encoded binary. For splice insert, the out-of-network indicator must be enabled. For time signal markers, the segmentation type ID, located inside the segmentation UPID, must be a cue-out value recognized by AWS Elemental MediaTailor. The ad avail starts at the event start and lasts for the event duration, if one is specified, or until the next event starts.

  The following example shows an event designated as an ad avail using splice insert markers. The duration for this ad avail is the event's duration.

```
 <Period start="PT444806.040S" id="123586" duration="PT15.000S">
   <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
     <Event duration="1350000">
       <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="180832" tier="4095">
         <scte35:SpliceInsert spliceEventId="4026531855"
spliceEventCancelIndicator="false" outOfNetworkIndicator="true"
spliceImmediateFlag="false" uniqueProgramId="1" availNum="1" availsExpected="1">
           <scte35:Program><scte35:SpliceTime ptsTime="5672624400"/></scte35:Program>
           <scte35:BreakDuration autoReturn="true" duration="1350000"/>
         </scte35:SpliceInsert>
       </scte35:SpliceInfoSection>
     </Event>
   </EventStream>
```

```
    <AdaptationSet mimeType="video/mp4"
        ...
    </AdaptationSet>
  </Period>
```

- Ad avails must have the same `AdaptationSet` and `Representation` settings as content streams. AWS Elemental MediaTailor uses these settings to transcode the ads to match the content stream, for smooth switching between the two.

After you configure the stream, note the content origin URL prefix for the DASH manifest. You need it to create the configuration in AWS Elemental MediaTailor, later in this tutorial.

# Step 3: Configure ADS Request URL and Query Parameters

To determine the query parameters that the ADS requires, generate an ad tag URL from the ADS. This URL acts as a template for requests to the ADS, and consists of the following:

- Static values
- Values generated by AWS Elemental MediaTailor (denoted by `session` or `avail` query parameters)
- Values generated by players, obtained from the client application (denoted by `player_params.` query parameters)

### Example Ad tag URL from an ADS

```
https://my.ads.com/ad?
output=vast&content_id=12345678&playerSession=[session.id]&cust_params=[player_params.cust_params]
```

Where:

- **output** and **content_id** are static values
- **playerSession=[session.id]** is a dynamic value provided by AWS Elemental MediaTailor. The value of **[session.id]** changes for each player session and results in a different URL for the VAST request for each session.
- **cust_params** are player-supplied dynamic values

The master manifest request from the player must provide key-value pairs that correspond to the `player_params.` query parameters in the ADS request URL. For more information about configuring key-value pairs in the request to AWS Elemental MediaTailor, see Dynamic Ad Variables in AWS Elemental MediaTailor (p. 60).

Enter the configured "template" URL when you create the origin server/ADS mapping in MediaTailor, in Step 4: Create a Configuration (p. 13).

### Testing

You can use a static VAST response from your ADS for testing purposes. Ideally, the VAST response returns a mezzanine quality MP4 rendition that AWS Elemental MediaTailor can transcode. If the response from the ADS contains multiple playback renditions, MediaTailor picks the highest quality and resolution MP4 rendition and sends it to the transcoder.

# Step 4: Create a Configuration

The AWS Elemental MediaTailor configuration holds mapping information for the origin server and ADS.

**To create a configuration (console)**

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. On the **Configurations** page, choose **Create configuration**.
3. In the **Configuration** section at the bottom of the page, for **Configuration name**, enter a unique name that describes the configuration. The name is the primary identifier for the configuration. The maximum length allowed is 512 characters.
4. For **Video content source**, enter the URL prefix for the HLS master manifest or DASH manifest for this stream, minus the asset ID. For example, if the master manifest URL is `http://origin-server.com/a/master.m3u8`, you would enter `http://origin-server.com/a/`. Alternatively, you can enter a shorter prefix such as `http://origin-server.com`, but then you must include the `/a/` in the asset ID in the player request for content. The maximum length is 512 characters.

   **Note**
   If your content origin uses HTTPS, its certificate must be from a well-known certificate authority. (It can't be a self-signed certificate.) Otherwise, AWS Elemental MediaTailor fails to connect to the content origin and can't serve manifests in response to player requests.
5. For **Ad decision server**, enter the URL for your ADS. This is either the URL with variables as described in Step 3: Configure ADS Request URL and Query Parameters (p. 12), or the static VAST URL that you are using for testing purposes. The maximum length is 25,000 characters.

   **Note**
   If your ADS uses HTTPS, its certificate must be from a well-known certificate authority. (It can't be a self-signed certificate.) The same is true for mezzanine ad URLs returned by the ADS. Otherwise, MediaTailor fails to retrieve and stitch ads into the manifests from the content origin.
6. (Optional as needed for DASH) For **Location**, choose **DISABLED** if you have CDN routing rules set up for accessing MediaTailor manifests and you are either using client-side reporting or your players support sticky HTTP redirects.

   For more information about the **Location** feature, see the section called “Location Feature” (p. 47).
7. (Optional) If your origin server produces single-period DASH manifests, choose **DASH mpd manifest origin type**, and then choose **SINGLE_PERIOD**. By default, MediaTailor handles DASH manifests as multi-period manifests. For more information, see the section called “DASH .mpd Manifests” (p. 21).
8. Choose **Create configuration**.

   AWS Elemental MediaTailor displays the new configuration on the **Configurations** page.

# Step 5: Test the Configuration

After you save the configuration, test the stream using a URL in the appropriate format for your streaming protocol:

- Example: HLS

```
playback-endpoint/v1/master/hashed-account-id/origin-id/master.m3u8
```

- Example: DASH

```
playback-endpoint/v1/dash/hashed-account-id/origin-id/manifest.mpd
```

Where:

- `playback-endpoint` is the unique playback endpoint that AWS Elemental MediaTailor generated when the configuration was created.

  Example

  ```
  https://bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com
  ```

- `hashed-account-id` is your AWS account ID.

  Example

  ```
  AKIAIOSFODNN7EXAMPLE
  ```

- `origin-id` is the name that you gave when creating the configuration.

  Example

  ```
  myOrigin
  ```

- `master.m3u8` or `manifest.mpd` is the name of the manifest from the test stream plus its file extension. Define this so that you get a fully identified manifest when you append this to the video content source that you configured in the section called “Step 4: Create a Configuration” (p. 13).

Using the values from the preceding examples, the full URLs are the following.

- Example: HLS

  ```
  https://bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com/v1/master/
  AKIAIOSFODNN7EXAMPLE/myOrigin/master.m3u8
  ```

- Example: DASH

  ```
  https://bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com/v1/dash/
  AKIAIOSFODNN7EXAMPLE/myOrigin/manifest.mpd
  ```

You can test the stream using one of the following methods.

- As shown in the preceding example, enter the URL in a standalone player.

- Test the stream in your own player environment.

# Step 6: Send the Playback Request to AWS Elemental MediaTailor

Configure the downstream player or CDN to send playback requests to the configuration's playback endpoint provided from AWS Elemental MediaTailor. Any player-defined dynamic variables that you used

in the ADS request URL in Step 3: Configure ADS Request URL and Query Parameters (p. 12) must be defined in the manifest request from the player.

**Example**

Assume your template ADS URL is the following.

```
https://my.ads.com/ad?
output=vast&content_id=12345678&playerSession=[session.id]&cust_params=[player_params.cust_params]
```

Then define `[player_params.cust_params]` in the player request by prefacing the key-value pair with `ads.`. AWS Elemental MediaTailor passes parameters that aren't preceded with `ads.` to the origin server instead of the ADS.

The player request URL is some variation of the following HLS and DASH examples.

```
https://bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com/v1/master/
AKIAIOSFODNN7EXAMPLE/myOrigin/master.m3u8?ads.cust_params=viewerinfo
```

```
https://bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com/v1/dash/
AKIAIOSFODNN7EXAMPLE/myOrigin/manifest.mpd?ads.cust_params=viewerinfo
```

When AWS Elemental MediaTailor receives the player request, it defines the player variables based on the information in the request. The resulting ADS request URL is some variation of this.

```
https://my.ads.com/ad?
output=vast&content_id=12345678&playerSession=<filled_in_session_id>&cust_params=viewerinfo
```

For more information about configuring key-value pairs to pass to the ADS, see Dynamic Ad Variables in AWS Elemental MediaTailor (p. 60).

# (Optional) Step 7: Monitor AWS Elemental MediaTailor Activity

Use Amazon CloudWatch and Amazon CloudWatch Logs to track AWS Elemental MediaTailor activity, such as the counts of requests, errors, and ad avails filled.

If this is your first time using CloudWatch with AWS Elemental MediaTailor, create an AWS Identity and Access Management (IAM) role to allow communication between the services.

**To allow AWS Elemental MediaTailor access to CloudWatch (console)**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
3. Choose the **Another AWS account** role type.
4. For **Account ID**, enter your AWS account ID.
5. Select **Require external ID** and enter **`midas`**. This option automatically adds a condition to the trust policy that allows the service to assume the role only if the request includes the correct `sts:ExternalID`.
6. Choose **Next: Permissions**.
7. Add a permissions policy that specifies what actions this role can complete. Select from one of the following options, and then choose **Next: Review**:

- **CloudWatchLogsFullAccess** to provide full access to Amazon CloudWatch Logs
- **CloudWatchFullAccess** to provide full access to Amazon CloudWatch

8. For **Role name**, enter **`MediaTailorLogger`**, and then choose **Create role**.
9. On the **Roles** page, select the role that you just created.
10. Edit the trust relationship to update the principal:

    1. On the role's **Summary** page, choose the **Trust relationship** tab.
    2. Choose **Edit trust relationship**.
    3. In the policy document, change the principal to the AWS Elemental MediaTailor service. It should look like this.

    ```
    "Principal": {
       "Service": "mediatailor.amazonaws.com"
    },
    ```

    The entire policy should read as follows.

    ```
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "mediatailor.amazonaws.com"
          },
          "Action": "sts:AssumeRole",
          "Condition": {
            "StringEquals": {
              "sts:ExternalId": "Midas"
            }
          }
        }
      ]
    }
    ```

    4. Choose **Update Trust Policy**.

# Step 8: Clean Up

To avoid extraneous charges, delete all unnecessary configurations.

### To delete a configuration (console)

1. On the AWS Elemental MediaTailor **Configurations** page, do one of the following:

   - Choose the **Configuration name** for the configuration that you want to delete.
   - In the **Configuration name** column, choose the radio button, and then choose **Delete**.

2. In the **Delete configuration** confirmation box, enter **`Delete`**, and then choose **Delete** again.

   AWS Elemental MediaTailor removes the configuration.

# AWS Elemental MediaTailor Manifest Handling

A manifest is the input to AWS Elemental MediaTailor from an upstream encoder. When MediaTailor receives a request for content playback, it manipulates the manifest and adds personalized content, tailored for the viewing session. This section describes how MediaTailor handles manifests. For information about ad handling and insertion, see Ad Behavior in AWS Elemental MediaTailor (p. 66).

**Topics**



## Ad Transcoding

When AWS Elemental MediaTailor stitches in the ads that are specified by the ad decision server (ADS) in the VAST response, it checks whether the ads have already been transcoded:

- If an ad has been transcoded, MediaTailor uses the ad in the ad avail.
- If it hasn't been transcoded, MediaTailor transcodes it for future use, but doesn't use it for the current request.

If there are multiple ads in the VAST response, AWS Elemental MediaTailor evaluates them sequentially and uses the ones that are already transcoded. If no ads are transcoded yet, MediaTailor plays the underlying content (or ad slate) instead of the ad.

## Alternate Audio and Subtitles

AWS Elemental MediaTailor supports input and output of multiple audio and WebVTT subtitle tracks.

### Audio

If your content contains alternate audio, AWS Elemental MediaTailor transcodes audio-only renditions of the ads to the alternate audio tracks for your content. This way, audio switching continues to work during ads. The service inserts the default audio from the ad and replicates it across your audio tracks during ad avails.

For ad transcoding to succeed, the audio sample rate must be from 16 to 320 kHz.

### Subtitles

Ad playback doesn't include subtitles. Instead, AWS Elemental MediaTailor inserts blank offsets for the webVTT sidecar files during ad avails.

For DASH, AWS Elemental MediaTailor supports in-band subtitles. MediaTailor currently does not support sideband subtitles.

# HLS .m3u8 Manifests

AWS Elemental MediaTailor supports `.m3u8` HLS manifests with an `EXT-X-VERSION` of `3` or higher for live streaming and video on demand (VOD). When MediaTailor encounters an ad avail, it attempts ad insertion or replacement, based on the type of content. If there aren't enough ads to fill the duration, for the remainder of the ad avail, MediaTailor displays the underlying content stream or the configured slate. For more information about HLS ad behavior based on content type (live or VOD), see Ad Behavior in AWS Elemental MediaTailor (p. 66).

The following sections provide more information about how MediaTailor handles HLS manifests.

**Topics**



## HLS Supported Ad Markers

AWS Elemental MediaTailor identifies ad avail boundaries in an HLS manifest ad markers in the input manifest. The following sections describe what markers MediaTailor uses.

### EXT-X-CUE-OUT and EXT-X-CUE-IN

This type of ad marker is the most common. The following examples show options for these cue markers.

```
#EXT-X-CUE-OUT:DURATION=120
...
#EXT-X-CUE-IN
```

```
#EXT-X-CUE-OUT:30.000
...
#EXT-X-CUE-IN
```

```
#EXT-X-CUE-OUT
...
#EXT-X-CUE-IN
```

### EXT-X-DATERANGE

With `EXT-X-DATERANGE` ad marker tags, you use `SCTE35-OUT` attributes to specify the timing of the ad avail.

> **Note**
> AWS Elemental MediaTailor ignores any `START-DATE` attributes that are provided for `EXT-X-DATERANGE` ad markers.

You can specify the ad avail in one of the following ways:

- `EXT-X-DATERANGE` tag with `SCTE35-OUT` and `DURATION` specifications.

  Example

```
#EXT-X-DATERANGE:ID="splice-6FFFFFF0",START-DATE="2019-01T00:15:00Z
\",DURATION=60.000,SCTE35-OUT=0xF
```

- Paired `EXT-X-DATERANGE` tags, the first with a `SCTE35-OUT` specification and the second with a `SCTE35-IN` specification.

  Example

```
#EXT-X-DATERANGE:ID="splice-6FFFFFF0",START-DATE="2019-01T00:15:00Z\",SCTE35-OUT=0xF
...
#EXT-X-DATERANGE:ID="splice-6FFFFFF0",START-DATE="2019-01T00:15:00Z\",SCTE35-IN=0xF
```

- A combination of the prior options. You specify an `EXT-X-DATERANGE` tag with `SCTE35-OUT` and `DURATION` specifications followed by an `EXT-X-DATERANGE` tag with a `SCTE35-IN` specification. In this case, MediaTailor uses the earliest cue-in setting from the two specifications.

  Example

```
#EXT-X-DATERANGE:ID="splice-6FFFFFF0",START-DATE="2019-01T00:15:00Z
\",DURATION=60.000,SCTE35-OUT=0xF
...
#EXT-X-DATERANGE:ID="splice-6FFFFFF0",START-DATE="2019-01T00:15:00Z\",SCTE35-IN=0xF
```

## EXT-X-SPLICEPOINT-SCTE35

You append the `EXT-X-SPLICEPOINT-SCTE35` ad marker tag with a SCTE-35 payload in base64-encoded binary. The decoded binary must provide a SCTE-35 `splice_info_section` containing the cue-out marker `0x34`, for provider placement opportunity start, and the cue-in marker `0x35`, for provider placement opportunity end.

The following example shows the splice point specification with base64-encoded binary payloads that specify the cue-out and cue-in markers.

```
#EXT-X-SPLICEPOINT-SCTE35:/DA9AAAAAAAAAP/wBQb+uYbZqwAnAiVDVUVJAAAKqX//
AAEjW4AMEU1EU05CMDAxMTMyMjE5M19ONAAAmXz5JA==
...
#EXT-X-SPLICEPOINT-SCTE35:/DA4AAAAAAAAAP/wBQb+tTeaawAiAiBDVUVJAAAKqH+/
DBFNRFNOQjAwMTEzMjIxOTJfTjUAAIiGK1s=
```

# HLS Manifest Tag Handling

This section describes how AWS Elemental MediaTailor manages tags in the personalized output manifest.

## EXT-X-CUE Tags

MediaTailor replaces `EXT-X-CUE-OUT`, `EXT-X-CUE-OUT-CONT`, and `EXT-X-CUE-IN` tags in the input manifest with `EXT-X-DISCONTINUITY` tags in the output manifest. The `DISCONTINUITY` tags mark the following boundaries:

- Where the main content transitions to an ad
- Where one ad transitions to another ad
- Where an ad transitions back to the main content

## EXT-X-DATERANGE Tags

MediaTailor passes through `EXT-X-DATERANGE` tags from the input manifest to the output manifest. MediaTailor also inserts `EXT-X-DISCONTINUITY` tags that correspond to the `DATERANGE` tags. The `DISCONTINUITY` tags mark the following boundaries:

- Where the main content transitions to an ad
- Where one ad transitions to another ad
- Where an ad transitions back to the main content

## EXT-X-KEY Tags

MediaTailor passes through `EXT-X-KEY` tags from the input manifest. These tags indicate that the main content is encrypted. Since ads aren't encrypted, MediaTailor inserts `EXT-X-KEY:METHOD=NONE` at the start of an ad avail. When playback returns to the main content, MediaTailor re-enables encryption by inserting the `EXT-X-KEY` tag with the `METHOD` value defined as the encryption type.

## Unrecognized Tags

MediaTailor passes through all unknown and custom tags from the input manifest to the output manifest.

# HLS Live Manifest Examples

The following example shows a valid live master manifest as input to AWS Elemental MediaTailor.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:BANDWIDTH=878612,RESOLUTION=640x360,CODECS="avc1.4D4029,mp4a.40.2"
scte35_1.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=2628628,RESOLUTION=1280x720,CODECS="avc1.4D4029,mp4a.40.2"
scte35_2.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=1128660,RESOLUTION=854x480,CODECS="avc1.4D4029,mp4a.40.2"
scte35_3.m3u8
```

The following example shows a media manifest as input to AWS Elemental MediaTailor. Note the `EXT-X-CUE-OUT` and `EXT-X-CUE-IN` tags that describe ad avail opportunities.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:4
#EXT-X-MEDIA-SEQUENCE:6719391
#EXTINF:4.000,
scte35_3_6719391.ts?m=1492714662
#EXTINF:3.533,
scte35_3_6719392.ts?m=1492714662
#EXT-OATCLS-SCTE35:/DAlAAALkmP0AP/wFAXwAlXbf+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXT-X-CUE-OUT:47.000
#EXTINF:0.467,
scte35_3_6719393.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=0.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:4.000,
scte35_3_6719394.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=4.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:4.000,
```

```
scte35_3_6719395.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=8.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:4.000,
scte35_3_6719396.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=12.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:4.000,
scte35_3_6719397.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=16.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:4.000,
scte35_3_6719398.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=20.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:4.000,
scte35_3_6719399.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=24.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:4.000,
scte35_3_6719400.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=28.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:4.000,
scte35_3_6719401.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=32.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:4.000,
scte35_3_6719402.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=36.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:4.000,
scte35_3_6719403.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=40.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:4.000,
scte35_3_6719404.ts?m=1492714662
#EXT-X-CUE-OUT-CONT:ElapsedTime=44.453,Duration=47.000,SCTE35=/DAlAAALkmP0AP/wFAXwAlXbf
+//4dg/yP4AQItwAAEBAQAAhT3BsQ==
#EXTINF:2.533,
scte35_3_6719405.ts?m=1492714662
#EXT-X-CUE-IN
#EXTINF:1.467,
scte35_3_6719406.ts?m=1492714662
```

# DASH .mpd Manifests

AWS Elemental MediaTailor supports `.mpd` live and video on demand (VOD) manifests that follow the guidelines for the DASH dynamic profile. MediaTailor accepts multi-period and single-period DASH-compliant manifest inputs, and delivers multi-period DASH-compliant manifest outputs.

Input manifests must have the following:

- At least one `Period` element with a `start` attribute.
- SCTE-35 event streams with splice info settings for either `splice insert` or `time signal`. The settings can be provided in clear XML or in base64-encoded binary.
- `Segment templates` with `segment timelines`.

For published manifests, MediaTailor requires that updates by the origin server leave the following unchanged:

- Period start times, specified in the `start` attribute.
- Values of `presentationTimeOffset` in the segment templates of the period representations.

As a best practice, give the ad avails the same `AdaptationSet` and `Representation` settings as the content stream periods. AWS Elemental MediaTailor uses these settings to transcode the ads to match the content stream, for smooth switching between the two.

The following sections provide more information about how MediaTailor handles DASH manifests.

**Topics**



# DASH Ad Markers

AWS Elemental MediaTailor identifies ad avails in a DASH manifest by splice insert and time signal cue-out markers, as follows:

- In a multi-period DASH manifest, a `Period` is considered an ad avail when the first `Event` in its event stream contains splice insert or time signal cue-out markers. In multi-period DASH, MediaTailor ignores all but the first event in a period.
- In a single-period DASH manifest, an `Event` is considered an ad avail when it contains splice insert or time signal cue-out markers.

By default, AWS Elemental MediaTailor manages DASH manifests as multi-period manifests. You can change your configuration to handle single-period DASH manifests from your origin server. For information, see the section called “Creating a Configuration” (p. 49).

You can provide ad markers in clear XML or in base64-encoded binary:

**Clear XML**

The event stream `schemeIdUri` must be set to `urn:scte:scte35:2013:xml`, and the event must have `scte35:SpliceInfoSection` markers containing one of the following:

- `scte35:SpliceInsert` with `outOfNetworkIndicator` set to `true`

  The following example shows this option, with the required markers in bold.

```
 <Period start="PT444806.040S" id="123586" duration="PT15.000S">
   <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
     <Event duration="1350000">
       <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="180832" tier="4095">
         <scte35:SpliceInsert spliceEventId="4026531855"
spliceEventCancelIndicator="false" outOfNetworkIndicator="true"
spliceImmediateFlag="false" uniqueProgramId="1" availNum="1" availsExpected="1">
           <scte35:Program><scte35:SpliceTime ptsTime="5672624400"/></scte35:Program>
           <scte35:BreakDuration autoReturn="true" duration="1350000"/>
         </scte35:SpliceInsert>
       </scte35:SpliceInfoSection>
     </Event>
```

- `scte35:TimeSignal` accompanied by `scte35:SegmentationDescriptor` `scte35:SegmentationUpid` with `segmentationTypeId` set to one of the following cue-out numbers:
  - 0x22 (start break)
  - 0x30 (provider advertisement start)
  - 0x32 (distributor advertisement start)
  - 0x34 (provider placement opportunity start)
  - 0x36 (distributor placement opportunity start)

  The following example shows this option, with the required markers in bold. The `segmentationTypeId` in this example is set to 52, equivalent to 0x34.

```
  <Period start="PT346530.250S" id="178443" duration="PT61.561S">
    <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
      <Event duration="5310000">
        <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="183003" tier="4095">
          <scte35:TimeSignal>
            <scte35:SpliceTime ptsTime="3442857000"/>
          </scte35:TimeSignal>
          <scte35:SegmentationDescriptor segmentationEventId="1414668"
 segmentationEventCancelIndicator="false" segmentationDuration="8100000">
            <scte35:DeliveryRestrictions webDeliveryAllowedFlag="false"
 noRegionalBlackoutFlag="false" archiveAllowedFlag="false" deviceRestrictions="3"/>
            <scte35:SegmentationUpid segmentationUpidType="12"
 segmentationUpidLength="2" segmentationTypeId="52" segmentNum="0"
 segmentsExpected="0">0100</scte35:SegmentationUpid>
          </scte35:SegmentationDescriptor>
        </scte35:SpliceInfoSection>
      </Event>
```

**Base64-encoded binary**

The event stream `schemeIdUri` must be set to `urn:scte:scte35:2014:xml+bin`, and the event must have `scte35:Signal` `scte35:Binary` that contains a base64-encoded binary. The decoded binary must provide a `splice_info_section` with the same set of information as the clear XML would provide in a `scte35:SpliceInfoSection` element. The command type must be either `splice_insert()` or `time_signal()`, and the additional settings must comply with those described previously for clear XML delivery.

The following example shows this option, with the required markers in bold.

```
  <Period start="PT444806.040S" id="123586" duration="PT15.000S">
    <EventStream schemeIdUri="urn:scte:scte35:2014:xml+bin" timescale="1">
      <Event presentationTime="1541436240" duration="24" id="29">
        <scte35:Signal xmlns="http://www.scte.org/schemas/35/2016">
          <scte35:Binary>/DAhAAAAAAAAAP/wEAUAAAHAf+9/fgAg9YDAAAAAAAA25aoh</Binary>
        </scte35:Signal>
      </Event>
      <Event presentationTime="1541436360" duration="24" id="30">
        <scte35:Signal xmlns="http://www.scte.org/schemas/35/2016">

 <scte35:Binary>QW5vdGhlciB0ZXN0IHN0cmluZyBmb3IgZW5jb2RpbmcgdG8gQmFzZTY0IGVuY29kZWQgYmluYXJ5Lg==</
Binary>
        </scte35:Signal>
      </Event>
```

The following is the decoded binary for the first event listed in the preceding example. The setting for `splice_command_type` is 5, which indicates `splice_insert`.

```
{
  "table_id": 252,
  "section_syntax_indicator": false,
  "private_indicator": false,
  "section_length": 33,
  "protocol_version": 0,
  "encrypted_packet": false,
  "encryption_algorithm": 0,
  "pts_adjustment": 0,
  "cw_index": 0,
  "tier": "0xFFF",
  "splice_command_length": 16,
  "splice_command_type": 5,
  "splice_command": {
    "splice_event_id": 448,
    "splice_event_cancel_indicator": false,
    "out_of_network_indicator": true,
    "program_splice_flag": true,
    "duration_flag": true,
    "splice_immediate_flag": false,
    "utc_splice_time": {
      "time_specified_flag": false,
      "pts_time": null
    },
    "component_count": 0,
    "components": null,
    "break_duration": {
      "auto_return": false,
      "duration": {
        "pts_time": 2160000,
        "wall_clock_seconds": 24.0,
        "wall_clock_time": "00:00:24:00000"
      }
    },
    "unique_program_id": 49152,
    "avail_num": 0,
    "avails_expected": 0
  },
  "splice_descriptor_loop_length": 0,
  "splice_descriptors": null,
  "Scte35Exception": {
    "parse_status": "SCTE-35 cue parsing completed with 0 errors.",
    "error_messages": [],
    "table_id": 252,
    "splice_command_type": 5
  }
}
```

For multi-period DASH manifests, AWS Elemental MediaTailor considers only the first `Event` in an event stream to determine ad replacement markers, and it ignores any additional `Event` markers in the stream. For single-period DASH manifests, MediaTailor considers each `Event`.

# DASH Ad Avail Duration

During playback, when AWS Elemental MediaTailor encounters an ad avail, it replaces some or all of the avail with ads. MediaTailor starts ad replacement at the beginning of the ad avail and includes ads as follows:

- If the ad avail specifies a duration, MediaTailor includes as many ads as it can fit inside the duration boundary, without overwriting content that follows.
- If no duration is provided, MediaTailor includes ads until it reaches the end of the ad avail. For multi-period manifests, this is the end of the period. For single-period manifests, this is the end of the event.

MediaTailor doesn't play ads past the end of the ad avail and, when it encounters the end, truncates the current ad instead of overwriting the content that follows.

**How AWS Elemental MediaTailor looks for the ad avail duration**

AWS Elemental MediaTailor searches for a duration setting in the following order:

1. `Event duration`
2. For splice insert, the `scte35:BreakDuration duration`
3. For time signal, the `scte35:SegmentationDescriptor segmentationDuration`

If AWS Elemental MediaTailor doesn't find any of these settings, it manages ad inclusion without a duration.

The following example shows an `Event` that has a `duration`.

```
  <Period start="PT444806.040S" id="123586" duration="PT15.000S">
    <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
      <Event duration="1350000">
        <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="180832" tier="4095">
          <scte35:SpliceInsert spliceEventId="4026531855"
 spliceEventCancelIndicator="false" outOfNetworkIndicator="true"
 spliceImmediateFlag="false" uniqueProgramId="1" availNum="1" availsExpected="1">
            <scte35:Program><scte35:SpliceTime ptsTime="5672624400"/></scte35:Program>
            <scte35:BreakDuration autoReturn="true" duration="1350000"/>
          </scte35:SpliceInsert>
        </scte35:SpliceInfoSection>
      </Event>
        ...
```

The following example shows ad avail with no duration specified. The `Event` has no `duration` and the `scte35:SpliceInsert` element doesn't contain a `scte35:BreakDuration` child element.

```
  <Period start="PT444836.720S" id="123597" duration="PT12.280S">
    <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
      <Event>
        <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="180832" tier="4095">
          <scte35:SpliceInsert spliceEventId="4026531856"
 spliceEventCancelIndicator="false" outOfNetworkIndicator="true"
 spliceImmediateFlag="false" uniqueProgramId="1" availNum="1" availsExpected="1">
            <scte35:Program><scte35:SpliceTime ptsTime="5675385600"/></scte35:Program>
          </scte35:SpliceInsert>
        </scte35:SpliceInfoSection>
      </Event>
      ...
```

# DASH Manifest Segment Numbering

MediaTailor supports media segments in `<SegmentTemplate>` that are defined using `<SegmentTimeline>` and the `media` attribute. You can specify the media segment list in the `media` attribute using either the `$Number$` identifier or the `$Time$` identifier.

The following example shows a `SegmentTemplate` with a `media` attribute setting that uses the `$Number$` identifier.

```
        <SegmentTemplate initialization="index_subtitles_4_0_init.mp4?
m=1532451703" media="index_subtitles_4_0_$Number$.mp4?m=1532451703"
 presentationTimeOffset="1062336677920" startNumber="2349899" timescale="90000">
```

```
          <SegmentTimeline>
            <S d="540540" r="2" t="1062338840080"/>
            <S d="69069" t="1062340461700"/>
          </SegmentTimeline>
        </SegmentTemplate>
```

The following example shows a `SegmentTemplate` with a `media` attribute setting that uses the `$Time$` identifier.

```
        <SegmentTemplate
 initialization="asset_720p_8000K_9_init.mp4" media="asset_720p_8000K_9_$Time$.mp4"
 startNumber="1" timescale="90000">
          <SegmentTimeline>
            <S d="180000" r="2" t="0"/>
            <S d="147000" t="540000"/>
          </SegmentTimeline>
        </SegmentTemplate>
```

# DASH Manifest Examples

This section lists examples for splice insert and time signal. Each example lists a manifest as received from the origin server and after AWS Elemental MediaTailor has personalized the manifest with ads.

**Topics**



## DASH Manifest Splice Insert Example

**DASH origin manifest example for splice insert**

The following example from an MPD manifest shows an ad avail in a manifest received by DASH from the content origin. This example uses the `scte35:SpliceInsert` markers with `outOfNetworkIndicator` set to `true`.

```
   <Period start="PT173402.036S" id="46041">
    <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
      <Event duration="9450000">
        <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="183265" tier="4095">
          <scte35:SpliceInsert spliceEventId="99" spliceEventCancelIndicator="false"
 outOfNetworkIndicator="true" spliceImmediateFlag="false" uniqueProgramId="1" availNum="1"
 availsExpected="1">
            <scte35:Program><scte35:SpliceTime ptsTime="7835775000"/></scte35:Program>
            <scte35:BreakDuration autoReturn="true" duration="9450000"/>
          </scte35:SpliceInsert>
        </scte35:SpliceInfoSection>
      </Event>
    </EventStream>
    <AdaptationSet mimeType="video/mp4" segmentAlignment="true" subsegmentAlignment="true"
 startWithSAP="1" subsegmentStartsWithSAP="1" bitstreamSwitching="true">
      <Representation id="1" width="640" height="360" frameRate="30/1" bandwidth="749952"
 codecs="avc1.4D4029">
        <SegmentTemplate timescale="30" media="index_video_1_0_$Number$.mp4?
m=1531257079" initialization="index_video_1_0_init.mp4?m=1531257079" startNumber="46042"
 presentationTimeOffset="5202061">
          <SegmentTimeline>
```

```
            <S t="5202061" d="115"/>
            <S t="5202176" d="120" r="4"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
      <Representation id="2" width="1280" height="720" frameRate="30/1" bandwidth="2499968"
 codecs="avc1.4D4029">
        <SegmentTemplate timescale="30" media="index_video_3_0_$Number$.mp4?
m=1531257079" initialization="index_video_3_0_init.mp4?m=1531257079" startNumber="46042"
 presentationTimeOffset="5202061">
          <SegmentTimeline>
            <S t="5202061" d="115"/>
            <S t="5202176" d="120" r="4"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
      <Representation id="3" width="1920" height="1080" frameRate="30/1"
 bandwidth="4499968" codecs="avc1.4D4029">
        <SegmentTemplate timescale="30" media="index_video_5_0_$Number$.mp4?
m=1531257079" initialization="index_video_5_0_init.mp4?m=1531257079" startNumber="46042"
 presentationTimeOffset="5202061">
          <SegmentTimeline>
            <S t="5202061" d="115"/>
            <S t="5202176" d="120" r="4"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
    </AdaptationSet>
    <AdaptationSet mimeType="audio/mp4" segmentAlignment="0" lang="eng">
      <Representation id="4" bandwidth="128858" audioSamplingRate="44100"
 codecs="mp4a.40.2">
        <SegmentTemplate timescale="44100" media="index_audio_2_0_$Number$.mp4?
m=1531257079" initialization="index_audio_2_0_init.mp4?m=1531257079" startNumber="46042"
 presentationTimeOffset="7647030507">
          <SegmentTimeline>
            <S t="7647030507" d="168959"/>
            <S t="7647199468" d="176127" r="1"/>
            <S t="7647551723" d="177151"/>
            <S t="7647728875" d="176127" r="1"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
      <Representation id="5" bandwidth="128858" audioSamplingRate="44100"
 codecs="mp4a.40.2">
        <SegmentTemplate timescale="44100" media="index_audio_4_0_$Number$.mp4?
m=1531257079" initialization="index_audio_4_0_init.mp4?m=1531257079" startNumber="46042"
 presentationTimeOffset="7647030507">
          <SegmentTimeline>
            <S t="7647030507" d="168959"/>
            <S t="7647199468" d="176127" r="1"/>
            <S t="7647551723" d="177151"/>
            <S t="7647728875" d="176127" r="1"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
      <Representation id="6" bandwidth="128858" audioSamplingRate="44100"
 codecs="mp4a.40.2">
        <SegmentTemplate timescale="44100" media="index_audio_6_0_$Number$.mp4?
m=1531257079" initialization="index_audio_6_0_init.mp4?m=1531257079" startNumber="46042"
 presentationTimeOffset="7647030507">
          <SegmentTimeline>
            <S t="7647030507" d="168959"/>
            <S t="7647199468" d="176127" r="1"/>
            <S t="7647551723" d="177151"/>
            <S t="7647728875" d="176127" r="1"/>
          </SegmentTimeline>
```

```
        </SegmentTemplate>
      </Representation>
    </AdaptationSet>
  </Period>
```

**DASH personalized response example for splice insert**

AWS Elemental MediaTailor personalizes the ad avails with advertising specifications. The personalizations reflect the viewer data that is received from the player and the advertising campaigns that are currently underway.

The following example shows an ad avail after MediaTailor personalizes it.

```
  <Period id="46041_1" start="PT48H10M2.036S">
    <BaseURL>http://cdnlocation.net/EXAMPLE_PRODUCT/</BaseURL>
    <AdaptationSet bitstreamSwitching="false" frameRate="30/1" mimeType="video/
mp4" segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
      <SegmentTemplate startNumber="1" timescale="90000"/>
      <Representation bandwidth="10000000" codecs="avc1.640028" height="1080" id="1"
 width="1920">
        <SegmentTemplate initialization="EXAMPLE_PRODUCT_1080p_10init.mp4"
 media="EXAMPLE_PRODUCT_1080p_10_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
      </Representation>
      <Representation bandwidth="4000000" codecs="avc1.64001f" height="720" id="2"
 width="1280">
        <SegmentTemplate initialization="EXAMPLE_PRODUCT_720p_9init.mp4"
 media="EXAMPLE_PRODUCT_720p_9_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
      </Representation>
      <Representation bandwidth="2500000" codecs="avc1.64001f" height="720" id="3"
 width="1280">
        <SegmentTemplate initialization="EXAMPLE_PRODUCT_720p_8init.mp4"
 media="EXAMPLE_PRODUCT_720p_8_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
      </Representation>
      <Representation bandwidth="2000000" codecs="avc1.64001f" height="540" id="4"
 width="960">
        <SegmentTemplate initialization="EXAMPLE_PRODUCT_540p_7init.mp4"
 media="EXAMPLE_PRODUCT_540p_7_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
      </Representation>
      <Representation bandwidth="1350000" codecs="avc1.64001e" height="396" id="5"
 width="704">
        <SegmentTemplate initialization="EXAMPLE_PRODUCT_396p_6init.mp4"
 media="EXAMPLE_PRODUCT_396p_6_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
      </Representation>
      <Representation bandwidth="900000" codecs="avc1.64001e" height="396" id="6"
 width="704">
        <SegmentTemplate initialization="EXAMPLE_PRODUCT_396p_5init.mp4"
 media="EXAMPLE_PRODUCT_396p_5_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
      </Representation>
      <Representation bandwidth="600000" codecs="avc1.64001e" height="396" id="7"
 width="704">
```

```
        <SegmentTemplate initialization="EXAMPLE_PRODUCT_396p_4init.mp4"
 media="EXAMPLE_PRODUCT_396p_4_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
      </Representation>
      <Representation bandwidth="450000" codecs="avc1.640016" height="288" id="8"
 width="512">
        <SegmentTemplate initialization="EXAMPLE_PRODUCT_288p_3init.mp4"
 media="EXAMPLE_PRODUCT_288p_3_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
      </Representation>
      <Representation bandwidth="300000" codecs="avc1.640016" height="288" id="9"
 width="512">
        <SegmentTemplate initialization="EXAMPLE_PRODUCT_288p_2init.mp4"
 media="EXAMPLE_PRODUCT_288p_2_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
      </Representation>
      <Representation bandwidth="200000" codecs="avc1.640016" height="288" id="10"
 width="512">
        <SegmentTemplate initialization="EXAMPLE_PRODUCT_288p_1init.mp4"
 media="EXAMPLE_PRODUCT_288p_1_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
      </Representation>
    </AdaptationSet>
    <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
      <SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a1_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a1_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"/>
      <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
 id="11"><SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a1_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a1_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"><SegmentTimeline><S d="96000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate></Representation>
    </AdaptationSet>
    <AdaptationSet lang="enm" mimeType="audio/mp4" segmentAlignment="0">
      <SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a2_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a2_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"/>
      <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
 id="12"><SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a2_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a2_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"><SegmentTimeline><S d="96000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate></Representation>
    </AdaptationSet>
    <AdaptationSet lang="por" mimeType="audio/mp4" segmentAlignment="0">
      <SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a3_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a3_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"/>
      <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
 id="13"><SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a3_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a3_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"><SegmentTimeline><S d="96000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate></Representation>
    </AdaptationSet>
    <AdaptationSet lang="spa" mimeType="audio/mp4" segmentAlignment="0">
      <SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a4_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a4_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"/>
      <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
 id="14"><SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a4_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a4_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"><SegmentTimeline><S d="96000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate></Representation>
```

```
    </AdaptationSet>
  </Period>
```

## DASH Manifest Time Signal Example

**DASH origin manifest example for time signal**

The following example shows an ad avail in a manifest received by DASH from the content origin. The following example shows the `scte35:TimeSignal` markers.

```
  <Period start="PT346530.250S" id="178443" duration="PT61.561S">
    <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
      <Event  duration="5310000">
        <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="183003" tier="4095">
          <scte35:TimeSignal>
            <scte35:SpliceTime ptsTime="3442857000"/>
          </scte35:TimeSignal>
          <scte35:SegmentationDescriptor segmentationEventId="1414668"
 segmentationEventCancelIndicator="false" segmentationDuration="8100000">
            <scte35:DeliveryRestrictions webDeliveryAllowedFlag="false"
 noRegionalBlackoutFlag="false" archiveAllowedFlag="false" deviceRestrictions="3"/>
            <scte35:SegmentationUpid segmentationUpidType="12" segmentationUpidLength="2"
 segmentationTypeId="52" segmentNum="0" segmentsExpected="0">0100</scte35:SegmentationUpid>
          </scte35:SegmentationDescriptor>
        </scte35:SpliceInfoSection>
      </Event>
    </EventStream>
    <AdaptationSet mimeType="video/mp4" segmentAlignment="true" subsegmentAlignment="true"
 startWithSAP="1" subsegmentStartsWithSAP="1" bitstreamSwitching="true">
      <Representation id="1" width="960" height="540" frameRate="30000/1001"
 bandwidth="1000000" codecs="avc1.4D401F">
        <SegmentTemplate timescale="30000" media="index_video_1_0_$Number$.mp4?
m=1528475245" initialization="index_video_1_0_init.mp4?m=1528475245" startNumber="178444"
 presentationTimeOffset="10395907501">
          <SegmentTimeline>
            <S t="10395907501" d="60060" r="29"/>
            <S t="10397709301" d="45045"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
    </AdaptationSet>
    <AdaptationSet mimeType="audio/mp4" segmentAlignment="0" lang="eng">
      <Representation id="2" bandwidth="96964" audioSamplingRate="48000"
 codecs="mp4a.40.2">
        <SegmentTemplate timescale="48000" media="index_audio_2_0_$Number$.mp4?
m=1528475245" initialization="index_audio_2_0_init.mp4?m=1528475245" startNumber="178444"
 presentationTimeOffset="16633452001">
          <SegmentTimeline>
            <S t="16633452289" d="96256" r="3"/>
            <S t="16633837313" d="95232"/>
            <S t="16633932545" d="96256" r="4"/>
            <S t="16634413825" d="95232"/>
            <S t="16634509057" d="96256" r="5"/>
            <S t="16635086593" d="95232"/>
            <S t="16635181825" d="96256" r="4"/>
            <S t="16635663105" d="95232"/>
            <S t="16635758337" d="96256" r="5"/>
            <S t="16636335873" d="71680"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
    </AdaptationSet>
  </Period>
```

**DASH personalized response example for time signal**

AWS Elemental MediaTailor personalizes the ad avails with advertising specifications. The personalizations reflect the viewer data that is received from the player and the advertising campaigns that are currently underway.

The following example shows an ad avail after AWS Elemental MediaTailor personalizes it.

```
  <Period id="178443_1" start="PT96H15M30.25S">
    <BaseURL>http://d2gh0tfpz97e4o.cloudfront.net/nbc_fallback_2/</BaseURL>
    <AdaptationSet bitstreamSwitching="false" frameRate="30/1" mimeType="video/
mp4" segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
      <SegmentTemplate startNumber="1" timescale="90000"/>
      <Representation bandwidth="10000000" codecs="avc1.640028" height="1080" id="1"
 width="1920">
        <SegmentTemplate initialization="nbc_fallback_ad_2_1080p_10init.mp4"
 media="nbc_fallback_ad_2_1080p_10_$Number%09d$.mp4" startNumber="1" timescale="90000">
          <SegmentTimeline>
            <S d="180000" r="13" t="0"/>
            <S d="176940" t="2520000"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
      <Representation bandwidth="4000000" codecs="avc1.64001f" height="720" id="2"
 width="1280">
        <SegmentTemplate initialization="nbc_fallback_ad_2_720p_9init.mp4"
 media="nbc_fallback_ad_2_720p_9_$Number%09d$.mp4" startNumber="1" timescale="90000">
          <SegmentTimeline>
            <S d="180000" r="13" t="0"/>
            <S d="176940" t="2520000"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
      <Representation bandwidth="2500000" codecs="avc1.64001f" height="720" id="3"
 width="1280">
        <SegmentTemplate initialization="nbc_fallback_ad_2_720p_8init.mp4"
 media="nbc_fallback_ad_2_720p_8_$Number%09d$.mp4" startNumber="1" timescale="90000">
          <SegmentTimeline>
            <S d="270000" r="8" t="0"/>
            <S d="266940" t="2430000"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
      <Representation bandwidth="2000000" codecs="avc1.64001f" height="540" id="4"
 width="960">
        <SegmentTemplate initialization="nbc_fallback_ad_2_540p_7init.mp4"
 media="nbc_fallback_ad_2_540p_7_$Number%09d$.mp4" startNumber="1" timescale="90000">
          <SegmentTimeline>
            <S d="360000" r="6" t="0"/>
            <S d="176940" t="2520000"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
      <Representation bandwidth="1350000" codecs="avc1.64001e" height="396" id="5"
 width="704">
        <SegmentTemplate initialization="nbc_fallback_ad_2_396p_6init.mp4"
 media="nbc_fallback_ad_2_396p_6_$Number%09d$.mp4" startNumber="1" timescale="90000">
          <SegmentTimeline>
            <S d="360000" r="6" t="0"/>
            <S d="176940" t="2520000"/>
          </SegmentTimeline>
```

```
      </SegmentTemplate>
    </Representation>
    <Representation bandwidth="900000" codecs="avc1.64001e" height="396" id="6"
width="704">
      <SegmentTemplate initialization="nbc_fallback_ad_2_396p_5init.mp4"
media="nbc_fallback_ad_2_396p_5_$Number%09d$.mp4" startNumber="1" timescale="90000">
        <SegmentTimeline>
          <S d="360000" r="6" t="0"/>
          <S d="176940" t="2520000"/>
        </SegmentTimeline>
      </SegmentTemplate>
    </Representation>
    <Representation bandwidth="600000" codecs="avc1.64001e" height="396" id="7"
width="704">
      <SegmentTemplate initialization="nbc_fallback_ad_2_396p_4init.mp4"
media="nbc_fallback_ad_2_396p_4_$Number%09d$.mp4" startNumber="1" timescale="90000">
        <SegmentTimeline>
          <S d="360000" r="6" t="0"/>
          <S d="176940" t="2520000"/>
        </SegmentTimeline>
      </SegmentTemplate>
    </Representation>
    <Representation bandwidth="450000" codecs="avc1.640016" height="288" id="8"
width="512">
      <SegmentTemplate initialization="nbc_fallback_ad_2_288p_3init.mp4"
media="nbc_fallback_ad_2_288p_3_$Number%09d$.mp4" startNumber="1" timescale="90000">
        <SegmentTimeline>
          <S d="360000" r="6" t="0"/>
          <S d="176940" t="2520000"/>
        </SegmentTimeline>
      </SegmentTemplate>
    </Representation>
    <Representation bandwidth="300000" codecs="avc1.640016" height="288" id="9"
width="512">
      <SegmentTemplate initialization="nbc_fallback_ad_2_288p_2init.mp4"
media="nbc_fallback_ad_2_288p_2_$Number%09d$.mp4" startNumber="1" timescale="90000">
        <SegmentTimeline>
          <S d="360000" r="6" t="0"/>
          <S d="176940" t="2520000"/>
        </SegmentTimeline>
      </SegmentTemplate>
    </Representation>
    <Representation bandwidth="200000" codecs="avc1.640016" height="288" id="10"
width="512">
      <SegmentTemplate initialization="nbc_fallback_ad_2_288p_1init.mp4"
media="nbc_fallback_ad_2_288p_1_$Number%09d$.mp4" startNumber="1" timescale="90000">
        <SegmentTimeline>
          <S d="180000" r="13" t="0"/>
          <S d="176940" t="2520000"/>
        </SegmentTimeline>
      </SegmentTemplate>
    </Representation>
  </AdaptationSet>
  <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
    <SegmentTemplate initialization="nbc_fallback_ad_2_audio_aac_a1_128kinit.mp4"
media="nbc_fallback_ad_2_audio_aac_a1_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000"/>
    <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
id="11">
      <SegmentTemplate initialization="nbc_fallback_ad_2_audio_aac_a1_128kinit.mp4"
media="nbc_fallback_ad_2_audio_aac_a1_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000">
        <SegmentTimeline>
          <S d="96000" r="13" t="0"/>
          <S d="94368" t="1344000"/>
        </SegmentTimeline>
```

```
      </SegmentTemplate>
    </Representation>
  </AdaptationSet>
  <AdaptationSet lang="enm" mimeType="audio/mp4" segmentAlignment="0">
    <SegmentTemplate initialization="nbc_fallback_ad_2_audio_aac_a2_128kinit.mp4"
media="nbc_fallback_ad_2_audio_aac_a2_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000"/>
    <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
id="12">
      <SegmentTemplate initialization="nbc_fallback_ad_2_audio_aac_a2_128kinit.mp4"
media="nbc_fallback_ad_2_audio_aac_a2_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000">
        <SegmentTimeline>
          <S d="96000" r="13" t="0"/>
          <S d="94368" t="1344000"/>
        </SegmentTimeline>
      </SegmentTemplate>
    </Representation>
  </AdaptationSet>
  <AdaptationSet lang="por" mimeType="audio/mp4" segmentAlignment="0">
    <SegmentTemplate initialization="nbc_fallback_ad_2_audio_aac_a3_128kinit.mp4"
media="nbc_fallback_ad_2_audio_aac_a3_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000"/>
    <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
id="13">
      <SegmentTemplate initialization="nbc_fallback_ad_2_audio_aac_a3_128kinit.mp4"
media="nbc_fallback_ad_2_audio_aac_a3_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000">
        <SegmentTimeline>
          <S d="96000" r="13" t="0"/>
          <S d="94368" t="1344000"/>
        </SegmentTimeline>
      </SegmentTemplate>
    </Representation>
  </AdaptationSet>
  <AdaptationSet lang="spa" mimeType="audio/mp4" segmentAlignment="0">
    <SegmentTemplate initialization="nbc_fallback_ad_2_audio_aac_a4_128kinit.mp4"
media="nbc_fallback_ad_2_audio_aac_a4_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000"/>
    <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
id="14">
      <SegmentTemplate initialization="nbc_fallback_ad_2_audio_aac_a4_128kinit.mp4"
media="nbc_fallback_ad_2_audio_aac_a4_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000">
        <SegmentTimeline>
          <S d="96000" r="13" t="0"/>
          <S d="94368" t="1344000"/>
        </SegmentTimeline>
      </SegmentTemplate>
    </Representation>
  </AdaptationSet>
 </Period>
```

## DASH Manifest Base64-encoded Binary Example with Single-Period Input

This example shows how AWS Elemental MediaTailor handles a manifest from an origin server that produces single-period manifests. You can indicate that your origin server produces single-period manifests in your MediaTailor configuration settings. MediaTailor produces multi-period DASH manifests, for both multi-period and single-period input manifests.

**DASH single-period origin manifest example for Base64-encoded binary**

The following example shows the input period's <EventStream>, with Base64-encoded binary ad avail events.

```
   <Period id="1" start="PT0S">
       <BaseURL>dash/</BaseURL>
       <EventStream schemeIdUri="urn:scte:scte35:2014:xml+bin" timescale="1">
           <Event presentationTime="1550252760" duration="24" id="136">
               <Signal xmlns="http://www.scte.org/schemas/35/2016">
                   <Binary>/DAhAAAAAAAAAP/wEAUAAACIf+9/fgAg9YDAAAAAAABiJjIs</Binary>
               </Signal>
           </Event>
           <Event presentationTime="1550252880" duration="24" id="137">
               <Signal xmlns="http://www.scte.org/schemas/35/2016">
                   <Binary>/DAhAAAAAAAAAP/wEAUAAACJf+9/fgAg9YDAAAAAAAC/KdNe</Binary>
               </Signal>
           </Event>
           <Event presentationTime="1550253000" duration="24" id="138">
               <Signal xmlns="http://www.scte.org/schemas/35/2016">
                   <Binary>/DAhAAAAAAAAAP/wEAUAAACKf+9/fgAg9YDAAAAAAADc+O1/</Binary>
               </Signal>
           </Event>
       </EventStream>
       <AdaptationSet...
       </AdaptationSet>
   </Period>
```

**DASH personalized response example for Base64-encoded binary, with single-period origin manifest configuration**

The following example reflects the personalization applied by AWS Elemental MediaTailor to the prior ad avails when the MediaTailor configuration indicates single-period DASH manifests from the origin server. MediaTailor produces a multi-period DASH manifest with personalizations that reflect the viewer data that is received from the player and the advertising campaigns that are currently underway.

```
  <Period id="0.0" start="PT0S">
       <BaseURL>dash/</BaseURL>
       <AdaptationSet audioSamplingRate="48000" codecs="mp4a.40.2" contentType="audio"
group="1" id="1" mimeType="audio/mp4" segmentAlignment="true" startWithSAP="1">
           <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="1"/>
           <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
           <Representation bandwidth="69000" id="audio=69000">
               <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="0" startNumber="1"
timescale="48000">
                   <SegmentTimeline>
                       <S d="48129" t="74412130844415"/>
                       <S d="48128" t="74412130892544"/>
                       <S d="48127" t="74412130940672"/>
                       <S d="48129" t="74412130988799"/>
                       <S d="48128" t="74412131036928"/>
                       <S d="47104" t="74412131085056"/>
                       <S d="48128" t="74412131132160"/>
                       <S d="48127" t="74412131180288"/>
                       <S d="48129" t="74412131228415"/>
                       <S d="48128" t="74412131276544"/>
                       <S d="48127" t="74412131324672"/>
                       <S d="48129" t="74412131372799"/>
                       <S d="48128" t="74412131420928"/>
                       <S d="47104" t="74412131469056"/>
                       <S d="48128" t="74412131516160"/>
                       <S d="48127" t="74412131564288"/>
                       <S d="48129" t="74412131612415"/>
```

```
                        <S d="48128" t="74412131660544"/>
                        <S d="48127" t="74412131708672"/>
                        <S d="48129" t="74412131756799"/>
                        <S d="48128" t="74412131804928"/>
                        <S d="47104" t="74412131853056"/>
                        <S d="48128" t="74412131900160"/>
                        <S d="48127" t="74412131948288"/>
                        <S d="48129" t="74412131996415"/>
                        <S d="48128" t="74412132044544"/>
                        <S d="48127" t="74412132092672"/>
                        <S d="48129" t="74412132140799"/>
                        <S d="48128" t="74412132188928"/>
                        <S d="47104" t="74412132237056"/>
                        <S d="48128" t="74412132284160"/>
                        <S d="48127" t="74412132332288"/>
                        <S d="48129" t="74412132380415"/>
                        <S d="48128" t="74412132428544"/>
                        <S d="48127" t="74412132476672"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
        <AdaptationSet codecs="avc1.64001F" contentType="video" group="2" height="720"
 id="2" mimeType="video/mp4" par="16:9" sar="1:1" segmentAlignment="true" startWithSAP="1"
 width="1280">
            <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
            <Representation bandwidth="700000" id="video=700000" scanType="progressive">
                <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
 media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="0" startNumber="1"
 timescale="90000">
                    <SegmentTimeline>
                        <S d="90000" r="34" t="139522745250000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
    <Period id="1550252760.0_1" start="PT430625H46M">
        <BaseURL>http://d2gh0tfpz97e4o.cloudfront.net/visitalps/</BaseURL>
        <AdaptationSet bitstreamSwitching="false" frameRate="30/1" mimeType="video/
mp4" segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
            <SegmentTemplate startNumber="1" timescale="90000"/>
            <Representation bandwidth="7500000" codecs="avc1.640028" height="1080" id="1"
 width="1920">
                <SegmentTemplate initialization="visitalps_1080p30_video_1080p_10init.mp4"
 media="visitalps_1080p30_video_1080p_10_$Number%09d$.mp4" startNumber="1"
 timescale="90000">
                    <SegmentTimeline>
                        <S d="180000" r="6" t="0"/>
                        <S d="86940" t="1260000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="3000000" codecs="avc1.64001f" height="720" id="2"
 width="1280">
                <SegmentTemplate initialization="visitalps_1080p30_video_720p_9init.mp4"
 media="visitalps_1080p30_video_720p_9_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180000" r="6" t="0"/>
                        <S d="86940" t="1260000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="1875000" codecs="avc1.64001f" height="720" id="3"
 width="1280">
```

```
                <SegmentTemplate initialization="visitalps_1080p30_video_720p_8init.mp4"
media="visitalps_1080p30_video_720p_8_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="270000" r="3" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="1500000" codecs="avc1.64001f" height="540" id="4"
width="960">
                <SegmentTemplate initialization="visitalps_1080p30_video_540p_7init.mp4"
media="visitalps_1080p30_video_540p_7_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="1012500" codecs="avc1.64001e" height="396" id="5"
width="704">
                <SegmentTemplate initialization="visitalps_1080p30_video_396p_6init.mp4"
media="visitalps_1080p30_video_396p_6_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="675000" codecs="avc1.64001e" height="396" id="6"
width="704">
                <SegmentTemplate initialization="visitalps_1080p30_video_396p_5init.mp4"
media="visitalps_1080p30_video_396p_5_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="450000" codecs="avc1.64001e" height="396" id="7"
width="704">
                <SegmentTemplate initialization="visitalps_1080p30_video_396p_4init.mp4"
media="visitalps_1080p30_video_396p_4_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="337500" codecs="avc1.640016" height="288" id="8"
width="512">
                <SegmentTemplate initialization="visitalps_1080p30_video_288p_3init.mp4"
media="visitalps_1080p30_video_288p_3_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="225000" codecs="avc1.640016" height="288" id="9"
width="512">
                <SegmentTemplate initialization="visitalps_1080p30_video_288p_2init.mp4"
media="visitalps_1080p30_video_288p_2_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
```

```
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="150000" codecs="avc1.640016" height="288" id="10"
width="512">
                <SegmentTemplate initialization="visitalps_1080p30_video_288p_1init.mp4"
media="visitalps_1080p30_video_288p_1_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180000" r="6" t="0"/>
                        <S d="86940" t="1260000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
        <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="visitalps_1080p30_audio_aac_128kinit.mp4"
media="visitalps_1080p30_audio_aac_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000"/>
            <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
id="11">
                <SegmentTemplate initialization="visitalps_1080p30_audio_aac_128kinit.mp4"
media="visitalps_1080p30_audio_aac_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000">
                    <SegmentTimeline>
                        <S d="96000" r="6" t="0"/>
                        <S d="46368" t="672000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
    <Period id="1550252760.0" start="PT430625H46M14.966S">
        <BaseURL>dash/</BaseURL>
        <EventStream schemeIdUri="urn:scte:scte35:2014:xml+bin" timescale="1">
            <Event duration="24" id="136" presentationTime="1550252760">
                <Signal xmlns="http://www.scte.org/schemas/35/2016">
                    <Binary>/DAhAAAAAAAAAP/wEAUAAACIf+9/fgAg9YDAAAAAAABiJjIs</Binary>
                </Signal>
            </Event>
        </EventStream>
        <AdaptationSet audioSamplingRate="48000" codecs="mp4a.40.2" contentType="audio"
group="1" id="1" mimeType="audio/mp4" segmentAlignment="true" startWithSAP="1">
            <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="1"/>
            <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
            <Representation bandwidth="69000" id="audio=69000">
                <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="74412133198368"
timescale="48000">
                    <SegmentTimeline>
                        <S d="48128" t="74412133196544"/>
                        <S d="48127" t="74412133244672"/>
                        <S d="48129" t="74412133292799"/>
                        <S d="48128" t="74412133340928"/>
                        <S d="47104" t="74412133389056"/>
                        <S d="48128" t="74412133436160"/>
                        <S d="48127" t="74412133484288"/>
                        <S d="48129" t="74412133532415"/>
                        <S d="48128" t="74412133580544"/>
                        <S d="48127" t="74412133628672"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
        <AdaptationSet codecs="avc1.64001F" contentType="video" group="2" height="720"
id="2" mimeType="video/mp4" par="16:9" sar="1:1" segmentAlignment="true" startWithSAP="1"
width="1280">
```

```
            <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
            <Representation bandwidth="700000" id="video=700000" scanType="progressive">
                <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="139522749746940"
timescale="90000">
                    <SegmentTimeline>
                        <S d="90000" r="9" t="139522749660000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
    <Period id="1550252784.0" start="PT430625H46M24S">
        <BaseURL>dash/</BaseURL>
        <AdaptationSet audioSamplingRate="48000" codecs="mp4a.40.2" contentType="audio"
group="1" id="1" mimeType="audio/mp4" segmentAlignment="true" startWithSAP="1">
            <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="1"/>
            <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
            <Representation bandwidth="69000" id="audio=69000">
                <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="74412133632000"
startNumber="60" timescale="48000">
                    <SegmentTimeline>
                        <S d="48129" t="74412133676799"/>
                        <S d="48128" t="74412133724928"/>
                        <S d="47104" t="74412133773056"/>
                        <S d="48128" t="74412133820160"/>
                        <S d="48127" t="74412133868288"/>
                        <S d="48129" t="74412133916415"/>
                        <S d="48128" t="74412133964544"/>
                        <S d="48127" t="74412134012672"/>
                        <S d="48129" t="74412134060799"/>
                        <S d="48128" t="74412134108928"/>
                        <S d="47104" t="74412134157056"/>
                        <S d="48128" t="74412134204160"/>
                        <S d="48127" t="74412134252288"/>
                        <S d="48129" t="74412134300415"/>
                        <S d="48128" t="74412134348544"/>
                        <S d="48127" t="74412134396672"/>
                        <S d="48129" t="74412134444799"/>
                        <S d="48128" t="74412134492928"/>
                        <S d="47104" t="74412134541056"/>
                        <S d="48128" t="74412134588160"/>
                        <S d="48127" t="74412134636288"/>
                        <S d="48129" t="74412134684415"/>
                        <S d="48128" t="74412134732544"/>
                        <S d="48127" t="74412134780672"/>
                        <S d="48129" t="74412134828799"/>
                        <S d="48128" t="74412134876928"/>
                        <S d="47104" t="74412134925056"/>
                        <S d="48128" t="74412134972160"/>
                        <S d="48127" t="74412135020288"/>
                        <S d="48129" t="74412135068415"/>
                        <S d="48128" t="74412135116544"/>
                        <S d="48127" t="74412135164672"/>
                        <S d="48129" t="74412135212799"/>
                        <S d="48128" t="74412135260928"/>
                        <S d="47104" t="74412135309056"/>
                        <S d="48128" t="74412135356160"/>
                        <S d="48127" t="74412135404288"/>
                        <S d="48129" t="74412135452415"/>
                        <S d="48128" t="74412135500544"/>
                        <S d="48127" t="74412135548672"/>
                        <S d="48129" t="74412135596799"/>
                        <S d="48128" t="74412135644928"/>
```

```
                    <S d="47104" t="74412135693056"/>
                    <S d="48128" t="74412135740160"/>
                    <S d="48127" t="74412135788288"/>
                    <S d="48129" t="74412135836415"/>
                    <S d="48128" t="74412135884544"/>
                    <S d="48127" t="74412135932672"/>
                    <S d="48129" t="74412135980799"/>
                    <S d="48128" t="74412136028928"/>
                    <S d="47104" t="74412136077056"/>
                    <S d="48128" t="74412136124160"/>
                    <S d="48127" t="74412136172288"/>
                    <S d="48129" t="74412136220415"/>
                    <S d="48128" t="74412136268544"/>
                    <S d="48127" t="74412136316672"/>
                    <S d="48129" t="74412136364799"/>
                    <S d="48128" t="74412136412928"/>
                    <S d="47104" t="74412136461056"/>
                    <S d="48128" t="74412136508160"/>
                    <S d="48127" t="74412136556288"/>
                    <S d="48129" t="74412136604415"/>
                    <S d="48128" t="74412136652544"/>
                    <S d="48127" t="74412136700672"/>
                    <S d="48129" t="74412136748799"/>
                    <S d="48128" t="74412136796928"/>
                    <S d="47104" t="74412136845056"/>
                    <S d="48128" t="74412136892160"/>
                    <S d="48127" t="74412136940288"/>
                    <S d="48129" t="74412136988415"/>
                    <S d="48128" t="74412137036544"/>
                    <S d="48127" t="74412137084672"/>
                    <S d="48129" t="74412137132799"/>
                    <S d="48128" t="74412137180928"/>
                    <S d="47104" t="74412137229056"/>
                    <S d="48128" t="74412137276160"/>
                    <S d="48127" t="74412137324288"/>
                    <S d="48129" t="74412137372415"/>
                    <S d="48128" t="74412137420544"/>
                    <S d="48127" t="74412137468672"/>
                    <S d="48129" t="74412137516799"/>
                    <S d="48128" t="74412137564928"/>
                    <S d="47104" t="74412137613056"/>
                    <S d="48128" t="74412137660160"/>
                    <S d="48127" t="74412137708288"/>
                    <S d="48129" t="74412137756415"/>
                    <S d="48128" t="74412137804544"/>
                    <S d="48127" t="74412137852672"/>
                    <S d="48129" t="74412137900799"/>
                    <S d="48128" t="74412137948928"/>
                    <S d="47104" t="74412137997056"/>
                    <S d="48128" t="74412138044160"/>
                    <S d="48127" t="74412138092288"/>
                    <S d="48129" t="74412138140415"/>
                    <S d="48128" t="74412138188544"/>
                    <S d="48127" t="74412138236672"/>
                </SegmentTimeline>
            </SegmentTemplate>
        </Representation>
    </AdaptationSet>
    <AdaptationSet codecs="avc1.64001F" contentType="video" group="2" height="720"
id="2" mimeType="video/mp4" par="16:9" sar="1:1" segmentAlignment="true" startWithSAP="1"
width="1280">
        <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
        <Representation bandwidth="700000" id="video=700000" scanType="progressive">
            <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="139522750560000"
startNumber="60" timescale="90000">
```

```
                    <SegmentTimeline>
                        <S d="90000" r="95" t="139522750560000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
    <Period id="1550252880.0_1" start="PT430625H48M">
        <BaseURL>http://d2gh0tfpz97e4o.cloudfront.net/visitalps/</BaseURL>
        <AdaptationSet bitstreamSwitching="false" frameRate="30/1" mimeType="video/
mp4" segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
            <SegmentTemplate startNumber="1" timescale="90000"/>
            <Representation bandwidth="7500000" codecs="avc1.640028" height="1080" id="1"
 width="1920">
                <SegmentTemplate initialization="visitalps_1080p30_video_1080p_10init.mp4"
 media="visitalps_1080p30_video_1080p_10_$Number%09d$.mp4" startNumber="1"
 timescale="90000">
                    <SegmentTimeline>
                        <S d="180000" r="6" t="0"/>
                        <S d="86940" t="1260000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="3000000" codecs="avc1.64001f" height="720" id="2"
 width="1280">
                <SegmentTemplate initialization="visitalps_1080p30_video_720p_9init.mp4"
 media="visitalps_1080p30_video_720p_9_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180000" r="6" t="0"/>
                        <S d="86940" t="1260000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="1875000" codecs="avc1.64001f" height="720" id="3"
 width="1280">
                <SegmentTemplate initialization="visitalps_1080p30_video_720p_8init.mp4"
 media="visitalps_1080p30_video_720p_8_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="270000" r="3" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="1500000" codecs="avc1.64001f" height="540" id="4"
 width="960">
                <SegmentTemplate initialization="visitalps_1080p30_video_540p_7init.mp4"
 media="visitalps_1080p30_video_540p_7_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="1012500" codecs="avc1.64001e" height="396" id="5"
 width="704">
                <SegmentTemplate initialization="visitalps_1080p30_video_396p_6init.mp4"
 media="visitalps_1080p30_video_396p_6_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="675000" codecs="avc1.64001e" height="396" id="6"
 width="704">
```

```
                <SegmentTemplate initialization="visitalps_1080p30_video_396p_5init.mp4"
media="visitalps_1080p30_video_396p_5_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="450000" codecs="avc1.64001e" height="396" id="7"
width="704">
                <SegmentTemplate initialization="visitalps_1080p30_video_396p_4init.mp4"
media="visitalps_1080p30_video_396p_4_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="337500" codecs="avc1.640016" height="288" id="8"
width="512">
                <SegmentTemplate initialization="visitalps_1080p30_video_288p_3init.mp4"
media="visitalps_1080p30_video_288p_3_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="225000" codecs="avc1.640016" height="288" id="9"
width="512">
                <SegmentTemplate initialization="visitalps_1080p30_video_288p_2init.mp4"
media="visitalps_1080p30_video_288p_2_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="150000" codecs="avc1.640016" height="288" id="10"
width="512">
                <SegmentTemplate initialization="visitalps_1080p30_video_288p_1init.mp4"
media="visitalps_1080p30_video_288p_1_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180000" r="6" t="0"/>
                        <S d="86940" t="1260000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
        <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="visitalps_1080p30_audio_aac_128kinit.mp4"
media="visitalps_1080p30_audio_aac_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000"/>
            <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
id="11">
                <SegmentTemplate initialization="visitalps_1080p30_audio_aac_128kinit.mp4"
media="visitalps_1080p30_audio_aac_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000">
                    <SegmentTimeline>
                        <S d="96000" r="6" t="0"/>
                        <S d="46368" t="672000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
```

```
   <Period id="1550252880.0" start="PT430625H48M14.966S">
       <BaseURL>dash/</BaseURL>
       <EventStream schemeIdUri="urn:scte:scte35:2014:xml+bin" timescale="1">
           <Event duration="24" id="137" presentationTime="1550252880">
               <Signal xmlns="http://www.scte.org/schemas/35/2016">
                   <Binary>/DAhAAAAAAAAAP/wEAUAAACJf+9/fgAg9YDAAAAAAAC/KdNe</Binary>
               </Signal>
           </Event>
       </EventStream>
       <AdaptationSet audioSamplingRate="48000" codecs="mp4a.40.2" contentType="audio"
group="1" id="1" mimeType="audio/mp4" segmentAlignment="true" startWithSAP="1">
           <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="1"/>
           <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
           <Representation bandwidth="69000" id="audio=69000">
               <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="74412138958368"
timescale="48000">
                   <SegmentTimeline>
                       <S d="48128" t="74412138956544"/>
                       <S d="48127" t="74412139004672"/>
                       <S d="48129" t="74412139052799"/>
                       <S d="48128" t="74412139100928"/>
                       <S d="47104" t="74412139149056"/>
                       <S d="48128" t="74412139196160"/>
                       <S d="48127" t="74412139244288"/>
                       <S d="48129" t="74412139292415"/>
                       <S d="48128" t="74412139340544"/>
                       <S d="48127" t="74412139388672"/>
                   </SegmentTimeline>
               </SegmentTemplate>
           </Representation>
       </AdaptationSet>
       <AdaptationSet codecs="avc1.64001F" contentType="video" group="2" height="720"
id="2" mimeType="video/mp4" par="16:9" sar="1:1" segmentAlignment="true" startWithSAP="1"
width="1280">
           <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
           <Representation bandwidth="700000" id="video=700000" scanType="progressive">
               <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="139522760546940"
timescale="90000">
                   <SegmentTimeline>
                       <S d="90000" r="9" t="139522760460000"/>
                   </SegmentTimeline>
               </SegmentTemplate>
           </Representation>
       </AdaptationSet>
   </Period>
   <Period id="1550252904.0" start="PT430625H48M24S">
       <BaseURL>dash/</BaseURL>
       <AdaptationSet audioSamplingRate="48000" codecs="mp4a.40.2" contentType="audio"
group="1" id="1" mimeType="audio/mp4" segmentAlignment="true" startWithSAP="1">
           <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="1"/>
           <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
           <Representation bandwidth="69000" id="audio=69000">
               <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="74412139392000"
startNumber="180" timescale="48000">
                   <SegmentTimeline>
                       <S d="48129" t="74412139436799"/>
                       <S d="48128" t="74412139484928"/>
                       <S d="47104" t="74412139533056"/>
                       <S d="48128" t="74412139580160"/>
                       <S d="48127" t="74412139628288"/>
                       <S d="48129" t="74412139676415"/>
```

```
<S d="48128" t="74412139724544"/>
<S d="48127" t="74412139772672"/>
<S d="48129" t="74412139820799"/>
<S d="48128" t="74412139868928"/>
<S d="47104" t="74412139917056"/>
<S d="48128" t="74412139964160"/>
<S d="48127" t="74412140012288"/>
<S d="48129" t="74412140060415"/>
<S d="48128" t="74412140108544"/>
<S d="48127" t="74412140156672"/>
<S d="48129" t="74412140204799"/>
<S d="48128" t="74412140252928"/>
<S d="47104" t="74412140301056"/>
<S d="48128" t="74412140348160"/>
<S d="48127" t="74412140396288"/>
<S d="48129" t="74412140444415"/>
<S d="48128" t="74412140492544"/>
<S d="48127" t="74412140540672"/>
<S d="48129" t="74412140588799"/>
<S d="48128" t="74412140636928"/>
<S d="47104" t="74412140685056"/>
<S d="48128" t="74412140732160"/>
<S d="48127" t="74412140780288"/>
<S d="48129" t="74412140828415"/>
<S d="48128" t="74412140876544"/>
<S d="48127" t="74412140924672"/>
<S d="48129" t="74412140972799"/>
<S d="48128" t="74412141020928"/>
<S d="47104" t="74412141069056"/>
<S d="48128" t="74412141116160"/>
<S d="48127" t="74412141164288"/>
<S d="48129" t="74412141212415"/>
<S d="48128" t="74412141260544"/>
<S d="48127" t="74412141308672"/>
<S d="48129" t="74412141356799"/>
<S d="48128" t="74412141404928"/>
<S d="47104" t="74412141453056"/>
<S d="48128" t="74412141500160"/>
<S d="48127" t="74412141548288"/>
<S d="48129" t="74412141596415"/>
<S d="48128" t="74412141644544"/>
<S d="48127" t="74412141692672"/>
<S d="48129" t="74412141740799"/>
<S d="48128" t="74412141788928"/>
<S d="47104" t="74412141837056"/>
<S d="48128" t="74412141884160"/>
<S d="48127" t="74412141932288"/>
<S d="48129" t="74412141980415"/>
<S d="48128" t="74412142028544"/>
<S d="48127" t="74412142076672"/>
<S d="48129" t="74412142124799"/>
<S d="48128" t="74412142172928"/>
<S d="47104" t="74412142221056"/>
<S d="48128" t="74412142268160"/>
<S d="48127" t="74412142316288"/>
<S d="48129" t="74412142364415"/>
<S d="48128" t="74412142412544"/>
<S d="48127" t="74412142460672"/>
<S d="48129" t="74412142508799"/>
<S d="48128" t="74412142556928"/>
<S d="47104" t="74412142605056"/>
<S d="48128" t="74412142652160"/>
<S d="48127" t="74412142700288"/>
<S d="48129" t="74412142748415"/>
<S d="48128" t="74412142796544"/>
<S d="48127" t="74412142844672"/>
```

```
                        <S d="48129" t="74412142892799"/>
                        <S d="48128" t="74412142940928"/>
                        <S d="47104" t="74412142989056"/>
                        <S d="48128" t="74412143036160"/>
                        <S d="48127" t="74412143084288"/>
                        <S d="48129" t="74412143132415"/>
                        <S d="48128" t="74412143180544"/>
                        <S d="48127" t="74412143228672"/>
                        <S d="48129" t="74412143276799"/>
                        <S d="48128" t="74412143324928"/>
                        <S d="47104" t="74412143373056"/>
                        <S d="48128" t="74412143420160"/>
                        <S d="48127" t="74412143468288"/>
                        <S d="48129" t="74412143516415"/>
                        <S d="48128" t="74412143564544"/>
                        <S d="48127" t="74412143612672"/>
                        <S d="48129" t="74412143660799"/>
                        <S d="48128" t="74412143708928"/>
                        <S d="47104" t="74412143757056"/>
                        <S d="48128" t="74412143804160"/>
                        <S d="48127" t="74412143852288"/>
                        <S d="48129" t="74412143900415"/>
                        <S d="48128" t="74412143948544"/>
                        <S d="48127" t="74412143996672"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
        <AdaptationSet codecs="avc1.64001F" contentType="video" group="2" height="720"
 id="2" mimeType="video/mp4" par="16:9" sar="1:1" segmentAlignment="true" startWithSAP="1"
 width="1280">
            <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
            <Representation bandwidth="700000" id="video=700000" scanType="progressive">
                <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
 media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="139522761360000"
 startNumber="180" timescale="90000">
                    <SegmentTimeline>
                        <S d="90000" r="95" t="139522761360000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
    <Period id="1550253000.0_1" start="PT430625H50M">
        <BaseURL>http://d2gh0tfpz97e4o.cloudfront.net/visitalps/</BaseURL>
        <AdaptationSet bitstreamSwitching="false" frameRate="30/1" mimeType="video/
mp4" segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
            <SegmentTemplate startNumber="1" timescale="90000"/>
            <Representation bandwidth="7500000" codecs="avc1.640028" height="1080" id="1"
 width="1920">
                <SegmentTemplate initialization="visitalps_1080p30_video_1080p_10init.mp4"
 media="visitalps_1080p30_video_1080p_10_$Number%09d$.mp4" startNumber="1"
 timescale="90000">
                    <SegmentTimeline>
                        <S d="180000" r="6" t="0"/>
                        <S d="86940" t="1260000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="3000000" codecs="avc1.64001f" height="720" id="2"
 width="1280">
                <SegmentTemplate initialization="visitalps_1080p30_video_720p_9init.mp4"
 media="visitalps_1080p30_video_720p_9_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180000" r="6" t="0"/>
```

```
                    <S d="86940" t="1260000"/>
                </SegmentTimeline>
            </SegmentTemplate>
        </Representation>
        <Representation bandwidth="1875000" codecs="avc1.64001f" height="720" id="3"
width="1280">
            <SegmentTemplate initialization="visitalps_1080p30_video_720p_8init.mp4"
media="visitalps_1080p30_video_720p_8_$Number%09d$.mp4" startNumber="1" timescale="90000">
                <SegmentTimeline>
                    <S d="270000" r="3" t="0"/>
                    <S d="266940" t="1080000"/>
                </SegmentTimeline>
            </SegmentTemplate>
        </Representation>
        <Representation bandwidth="1500000" codecs="avc1.64001f" height="540" id="4"
width="960">
            <SegmentTemplate initialization="visitalps_1080p30_video_540p_7init.mp4"
media="visitalps_1080p30_video_540p_7_$Number%09d$.mp4" startNumber="1" timescale="90000">
                <SegmentTimeline>
                    <S d="360000" r="2" t="0"/>
                    <S d="266940" t="1080000"/>
                </SegmentTimeline>
            </SegmentTemplate>
        </Representation>
        <Representation bandwidth="1012500" codecs="avc1.64001e" height="396" id="5"
width="704">
            <SegmentTemplate initialization="visitalps_1080p30_video_396p_6init.mp4"
media="visitalps_1080p30_video_396p_6_$Number%09d$.mp4" startNumber="1" timescale="90000">
                <SegmentTimeline>
                    <S d="360000" r="2" t="0"/>
                    <S d="266940" t="1080000"/>
                </SegmentTimeline>
            </SegmentTemplate>
        </Representation>
        <Representation bandwidth="675000" codecs="avc1.64001e" height="396" id="6"
width="704">
            <SegmentTemplate initialization="visitalps_1080p30_video_396p_5init.mp4"
media="visitalps_1080p30_video_396p_5_$Number%09d$.mp4" startNumber="1" timescale="90000">
                <SegmentTimeline>
                    <S d="360000" r="2" t="0"/>
                    <S d="266940" t="1080000"/>
                </SegmentTimeline>
            </SegmentTemplate>
        </Representation>
        <Representation bandwidth="450000" codecs="avc1.64001e" height="396" id="7"
width="704">
            <SegmentTemplate initialization="visitalps_1080p30_video_396p_4init.mp4"
media="visitalps_1080p30_video_396p_4_$Number%09d$.mp4" startNumber="1" timescale="90000">
                <SegmentTimeline>
                    <S d="360000" r="2" t="0"/>
                    <S d="266940" t="1080000"/>
                </SegmentTimeline>
            </SegmentTemplate>
        </Representation>
        <Representation bandwidth="337500" codecs="avc1.640016" height="288" id="8"
width="512">
            <SegmentTemplate initialization="visitalps_1080p30_video_288p_3init.mp4"
media="visitalps_1080p30_video_288p_3_$Number%09d$.mp4" startNumber="1" timescale="90000">
                <SegmentTimeline>
                    <S d="360000" r="2" t="0"/>
                    <S d="266940" t="1080000"/>
                </SegmentTimeline>
            </SegmentTemplate>
        </Representation>
        <Representation bandwidth="225000" codecs="avc1.640016" height="288" id="9"
width="512">
```

```
                <SegmentTemplate initialization="visitalps_1080p30_video_288p_2init.mp4"
media="visitalps_1080p30_video_288p_2_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="360000" r="2" t="0"/>
                        <S d="266940" t="1080000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="150000" codecs="avc1.640016" height="288" id="10"
width="512">
                <SegmentTemplate initialization="visitalps_1080p30_video_288p_1init.mp4"
media="visitalps_1080p30_video_288p_1_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180000" r="6" t="0"/>
                        <S d="86940" t="1260000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
        <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="visitalps_1080p30_audio_aac_128kinit.mp4"
media="visitalps_1080p30_audio_aac_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000"/>
            <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
id="11">
                <SegmentTemplate initialization="visitalps_1080p30_audio_aac_128kinit.mp4"
media="visitalps_1080p30_audio_aac_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000">
                    <SegmentTimeline>
                        <S d="96000" r="6" t="0"/>
                        <S d="46368" t="672000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
    <Period id="1550253000.0" start="PT430625H50M14.966S">
        <BaseURL>dash/</BaseURL>
        <EventStream schemeIdUri="urn:scte:scte35:2014:xml+bin" timescale="1">
            <Event duration="24" id="138" presentationTime="1550253000">
                <Signal xmlns="http://www.scte.org/schemas/35/2016">
                    <Binary>/DAhAAAAAAAAAP/wEAUAAACKf+9/fgAg9YDAAAAAAADc+O1/</Binary>
                </Signal>
            </Event>
        </EventStream>
        <AdaptationSet audioSamplingRate="48000" codecs="mp4a.40.2" contentType="audio"
group="1" id="1" mimeType="audio/mp4" segmentAlignment="true" startWithSAP="1">
            <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="1"/>
            <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
            <Representation bandwidth="69000" id="audio=69000">
                <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="74412144718368"
timescale="48000">
                    <SegmentTimeline>
                        <S d="48128" t="74412144716544"/>
                        <S d="48127" t="74412144764672"/>
                        <S d="48129" t="74412144812799"/>
                        <S d="48128" t="74412144860928"/>
                        <S d="47104" t="74412144909056"/>
                        <S d="48128" t="74412144956160"/>
                        <S d="48127" t="74412145004288"/>
                        <S d="48129" t="74412145052415"/>
                        <S d="48128" t="74412145100544"/>
                        <S d="48127" t="74412145148672"/>
                    </SegmentTimeline>
```

```
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
        <AdaptationSet codecs="avc1.64001F" contentType="video" group="2" height="720"
id="2" mimeType="video/mp4" par="16:9" sar="1:1" segmentAlignment="true" startWithSAP="1"
width="1280">
            <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
            <Representation bandwidth="700000" id="video=700000" scanType="progressive">
                <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="139522771346940"
timescale="90000">
                    <SegmentTimeline>
                        <S d="90000" r="9" t="139522771260000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
```

# DASH Location Feature

This section provides information about the location feature for DASH, which is enabled by default in AWS Elemental MediaTailor. Read this section if you create content delivery network (CDN) routing rules for accessing MediaTailor manifests. Also read this section if you use server-side reporting with players that don't support sticky HTTP redirects.

**What is the location feature?**

The location feature allows players that don't support sticky HTTP redirects to provide sticky behavior in their manifest update requests.

AWS Elemental MediaTailor uses sessionless initialization, and it requires sticky HTTP redirect behavior from its players. With server-side reporting, when the player makes a request for a manifest update to MediaTailor, the service issues a 302 temporary redirect, to direct the player to an endpoint for the personalized manifest. MediaTailor includes a session ID in the response, as a query parameter. The intent is for the player to follow the URL for the entirety of the session, but players that don't support sticky HTTP redirects drop the redirect and return to the original URL. When a player returns to the original URL, for each new request MediaTailor creates a new session rather than staying with the original session. This can cause the manifest to become corrupt.

The DASH specification provides a solution to this problem in the location feature, which is enabled by default in AWS Elemental MediaTailor configurations. When this feature is enabled, MediaTailor puts the absolute URL in the manifest `<Location>` tag. Players that don't support sticky HTTP redirects can use the URL provided in `<Location>` to request updates to the manifest.

**Do I need to disable the location feature in my configuration?**

The location feature overrides any CDN routing rules that you set up for accessing AWS Elemental MediaTailor manifests, so you might need to disable it. The location feature doesn't affect CDN caching of content or ad segments.

Find your situation in the following list to determine whether you need to disable the location feature for your configuration and how to handle it:

- If you don't have CDN routing rules set up for accessing AWS Elemental MediaTailor manifests, leave the location setting enabled.
- Otherwise, use the following rules:
  - If you either don't use server-side reporting or your players all support sticky HTTP redirects, disable the location feature. For information about how to do this on the console, see the section called “Creating a Configuration” (p. 49).

- Otherwise, contact AWS Support.

**Do I need to use the location feature?**

You need to use the location feature for players that don't support sticky HTTP redirects, for example, Shaka. Use the URL provided in the `<Location>` tag for all of your manifest update requests.

**Example**

Example URLs and example `<Location>` tag.

- **Example Example: Initial request URL**

```
https://b00f3e55c5cb4c1ea6dee499964bea92.mediatailor.us-east-1.amazonaws.com/v1/
dash/5ca4c1892b1f213a1247fad47b3e34c454a7d490/testLocationTag/index.mpd
```

- **Example Example: Redirected 302 response**

```
/v1/dash/5ca4c1892b1f213a1247fad47b3e34c454a7d490/testLocationTag/index.mpd?
aws.sessionId=0e5d9b45-ae97-49eb-901b-893d043e0aa6
```

- **Example Example: Location tag in a manifest**

```
<Location>https://b00f3e55c5cb4c1ea6dee499964bea92.mediatailor.us-east-1.amazonaws.com/
v1/dash/5ca4c1892b1f213a1247fad47b3e34c454a7d490/testLocationTag/index.mpd?
aws.sessionId=0e5d9b45-ae97-49eb-901b-893d043e0aa6</Location>
```

# Working with Configurations in AWS Elemental MediaTailor

A configuration is an object that you interact with in AWS Elemental MediaTailor. The configuration holds the mapping information for the origin server and the ad decision server (ADS). You can also define a default playback for MediaTailor to use when an ad isn't available or doesn't fill the entire ad avail.

If you use a content distribution network (CDN) with MediaTailor, you must set up the behavior rules in the CDN before you add CDN information to the configuration. For more information about setting up your CDN, see Integrating AWS Elemental MediaTailor and a CDN (p. 53).

**Topics**



## Creating a Configuration

Create a configuration to start receiving content streams and to provide an access point for downstream playback devices to request content.

You can use the AWS Elemental MediaTailor console, the AWS CLI, or the MediaTailor API to create a configuration. For information about creating a configuration through the AWS CLI or MediaTailor API, see the https://docs.aws.amazon.com/mediatailor/latest/apireference/.

When you're creating a configuration, do not put sensitive identifying information like customer account numbers into free-form fields such as the **Configuration name** field. This includes when you work with MediaTailor using the console, REST API, AWS CLI, or AWS SDKs. Any data that you enter into MediaTailor might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

### To add a configuration (console)

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. On the **Configurations** page, choose **Create configuration**.
3. Complete the configuration and additional configuration fields as described in the following topics:
   - Main Configuration Fields (p. 50)
   - Additional Configuration Fields (p. 50)
4. Choose **Create configuration**.

   AWS Elemental MediaTailor displays the new configuration in the table on the **Configurations** page.
5. (Optional, but recommended) You can use the configuration playback URLs to set up a CDN with AWS Elemental MediaTailor for manifests and reporting.

   For information about setting up a CDN for manifest and reporting requests, see Integrating AWS Elemental MediaTailor and a CDN (p. 53).

# Main Configuration Fields

When you create a configuration, you must complete at least the main configuration fields, described here.

**Configuration name**

Enter a unique name that describes the configuration. The name is the primary identifier for the configuration. The maximum length allowed is 512 characters.

**Video content source**

Enter the URL prefix for the manifest for this stream, minus the asset ID. The maximum length is 512 characters.

For example, the URL prefix `http://origin-server.com/a/` is valid for an HLS master manifest URL of `http://origin-server.com/a/master.m3u8` and for a DASH manifest URL of `http://origin-server.com/a/dash.mpd`. Alternatively, you can enter a shorter prefix such as `http://origin-server.com`, but the `/a/` must be included in the asset ID in the player request for content.

**Note**
If your content origin uses HTTPS, its certificate must be from a well-known certificate authority. (It can't be a self-signed certificate.) Otherwise, AWS Elemental MediaTailor fails to connect to the content origin and can't serve manifests in response to player requests.

**Ad decision server**

Enter the URL for your ad decision server (ADS). This is either the URL with variables as described in Step 3: Configure ADS Request URL and Query Parameters (p. 12), or the static VAST URL that you are using for testing purposes. The maximum length is 25,000 characters.

**Note**
If your ADS uses HTTPS, its certificate must be from a well-known certificate authority. (It can't be a self-signed certificate.) The same also applies to mezzanine ad URLs returned by the ADS. Otherwise, AWS Elemental MediaTailor can't retrieve and stitch ads into the manifests from the content origin.

# Additional Configuration Fields

Choose **Additional configuration** to complete optional fields, described here.

**Slate ad**

Enter the URL for a high-quality MP4 asset to transcode and use to fill in time that's not used by ads. AWS Elemental MediaTailor shows the slate to fill in gaps in media content. Configuring the slate is optional for non-VPAID configurations. For VPAID, you must configure a slate, which MediaTailor provides in the slots designated for dynamic ad content. The slate must be a high-quality MP4 asset that contains both audio and video. For more information, see Slate Management (p. 69).

**Note**
If the server that hosts your slate uses HTTPS, its certificate must be from a well-known certificate authority. (It can't be a self-signed certificate.) Otherwise, AWS Elemental MediaTailor can't retrieve and stitch the slate into the manifests from the content origin.

**Live pre-roll ad decision server**

To insert ads at the start of a live stream before the main content starts playback, enter the URL for the ad pre-roll from the ad decision server (ADS). This is either the URL with variables as described in Step 3: Configure ADS Request URL and Query Parameters (p. 12), or the static VAST URL that you are using for testing purposes. The maximum length is 25,000 characters.

**Note**
If your ADS uses HTTPS, its certificate must be from a well-known certificate authority. (It can't be a self-signed certificate.) The same also applies to mezzanine ad URLs returned by the ADS. Otherwise, AWS Elemental MediaTailor can't retrieve and stitch ads into the manifests from the content origin.

**Live pre-roll maximum allowed duration**

When you're inserting ads at the start of a live stream, enter the maximum allowed duration for the pre-roll ad avail. MediaTailor won't go over this duration when inserting ads. If the response from the ADS contains more ads than will fit in this duration, MediaTailor fills the avail with as many ads as possible, without going over the duration. For more details about how MediaTailor fills avails, see Live Content Ad Behavior (p. 68).

**CDN content segment prefix**

Enables AWS Elemental MediaTailor to create manifests with URLs to your CDN path for content segments. Before you do this step, set up a rule in your CDN to pull segments from your origin server. For **CDN content segment prefix**, enter the CDN prefix path.

For more information about integrating MediaTailor with a CDN, see CDN Integration with AWS Elemental MediaTailor (p. 53).

**CDN ad segment prefix**

Enables AWS Elemental MediaTailor to create manifests with URLs to your own CDN path for ad segments. By default, MediaTailor serves ad segments from an internal Amazon CloudFront distribution with default cache settings. Before you can complete the **CDN ad segment prefix** field, you must set up a rule in your CDN to pull ad segments from the following origin, like in the following example.

```
https://segments.mediatailor.<region>.amazonaws.com
```

For **CDN ad segment prefix**, enter the name of your CDN prefix in the configuration.

For more information about integrating MediaTailor with a CDN, see CDN Integration with AWS Elemental MediaTailor (p. 53).

**DASH mpd location**

(Optional as needed for DASH) Choose **DISABLED** if you have CDN routing rules set up for accessing MediaTailor manifests and you are either using client-side reporting or your players support sticky HTTP redirects.

For more information about the **Location** feature, see the section called “Location Feature” (p. 47).

**DASH origin manifest type**

If your origin server produces single-period DASH manifests, open the dropdown list and choose **SINGLE_PERIOD**. By default, MediaTailor handles DASH manifests as multi-period manifests. For more information, see the section called “DASH .mpd Manifests” (p. 21).

# Viewing a Configuration

You can view the configuration's current settings.

**To view a configuration**

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. On the **Configurations** page, choose the **Configuration name** for the configuration to view.

In addition to the values provided when the configuration was created, AWS Elemental MediaTailor displays the name of the configuration, playback endpoints, and relevant access URLs.

# Editing a Configuration

You can edit a configuration to update the origin server and ad decision server (ADS) mapping, or change how AWS Elemental MediaTailor interacts with a content distribution network (CDN).

**To edit a configuration**

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. On the **Configurations** page, choose the name of the configuration that you want to edit.
3. On the configuration details page, choose **Edit**, and then revise the configuration settings as needed. You can't edit the configuration name. For information about configuration attributes, see Creating a Configuration (p. 49).
4. Choose **Save**.

# Deleting a Configuration

You can delete a configuration to make it unavailable for playback.

**To delete a configuration**

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. On the **Configurations** page, do one of the following:
   - Choose the name of the configuration that you want to delete.
   - In the **Configuration name** column, choose the option next to the name, and then choose **Delete**.
3. In the **Delete** confirmation box, enter **`Delete`**, and then choose **Delete**.

# CDN Integration with AWS Elemental MediaTailor

We highly recommend that you use a content distribution network (CDN) such as Amazon CloudFront to improve the efficiency of the ad stitching workflow between AWS Elemental MediaTailor and your users. The benefits of a CDN include content and ad caching, consistent domain names across personalized manifests, and CDN DNS resolution.

When you use a CDN in the AWS Elemental MediaTailor workflow, the request and response flow is as follows:

1. The player requests a manifest from the CDN with MediaTailor as the manifest origin. The CDN forwards the request to MediaTailor.
2. MediaTailor personalizes the manifest and substitutes CDN domain names for the content and ad segment URL prefixes. MediaTailor sends the personalized manifest as a response to the CDN, which forwards it to the requesting player.
3. The player requests segments from the URLs that are provided in the manifest.
4. The CDN translates the segment URLs. It forwards content segment requests to the origin server and forwards ad requests to the Amazon CloudFront distribution where MediaTailor stores transcoded ads.
5. The origin server and MediaTailor respond with the requested segments, and playback begins.

The following sections describe how to configure AWS Elemental MediaTailor and the CDN to perform this flow.

## Integrating AWS Elemental MediaTailor and a CDN

The following steps show how to integrate AWS Elemental MediaTailor with your content distribution network (CDN). Depending on the CDN that you use, some terminology might differ from what is used in these steps.

### Step 1: (CDN) Create Routing Behaviors

In the CDN, create behaviors and rules that route content segment requests to the origin server and ad segment requests to AWS Elemental MediaTailor, as follows:

- Create one behavior that routes *content segment* requests to the *origin server*. Base this on a rule that uses a phrase to differentiate content segment requests from ad segment requests.

  For example, the CDN could route HLS player requests to `https://CDN_Hostname/subdir/content.ts` to the origin server path `http://origin.com/contentpath/subdir/content.ts` based on the keyword `subdir` in the request.

  For example, the CDN could route DASH player requests to `https://CDN_Hostname/subdir/content.mp4` to the origin server path `http://origin.com/contentpath/subdir/content.mp4` based on the keyword `subdir` in the request.

- (Optional) Create one behavior that routes *ad segment* requests to the internal Amazon CloudFront distribution where AWS Elemental MediaTailor stores transcoded ads. Base this on a rule that includes

a phrase to differentiate ad segment requests from content segment requests. This step is optional because AWS Elemental MediaTailor provides a default configuration.

AWS Elemental MediaTailor uses the following default Amazon CloudFront distributions for storing ads:

**Example Ad Segment Routing**

Pattern: `https://segments.mediatailor.<region>.amazonaws.com`

Example: `https://segments.mediatailor.eu-west-1.amazonaws.com`

# Step 2: (AWS Elemental MediaTailor) Create a Configuration with CDN Mapping

Create an AWS Elemental MediaTailor configuration that maps the domains of the CDN routing behaviors to the origin server and to the ad storage location. Enter the domain names in the configuration as follows:

- For **CDN content segment prefix**, enter the CDN domain from the behavior that you created to route content requests to the origin server. In the manifest, MediaTailor replaces the content segment URL prefix with the CDN domain.

  For example, consider the following settings.
  - **Video content source** in the MediaTailor configuration is `http://origin.com/contentpath/`
  - **CDN content segment prefix** is `https://CDN_Hostname/`

  For HLS, if the full content file path is `http://origin.com/contentpath/subdir/content.ts`, the content segment in the manifest served by MediaTailor is `https://CDN_Hostname/subdir/content.ts`.

  For DASH, if the full content file path is `http://origin.com/contentpath/subdir/content.mp4`, the content segment in the manifest served by MediaTailor is `https://CDN_Hostname/subdir/content.mp4`.
- For **CDN ad segment prefix**, enter the name of the CDN behavior that you created to route ad requests through your CDN. In the manifest, MediaTailor replaces the Amazon CloudFront distribution with the behavior name.

# Step 3: (CDN) Set up CDN for Manifest and Reporting Requests

Using a CDN for manifest and reporting requests gives you more functionality in your workflow.

For manifests, referencing a CDN in front of the manifest specification lets you use CDN features such as geofencing, and also lets you serve everything from your own domain name. For this path, do not cache the manifests because they are all personalized. Manifest specifications are `/v1/master` for HLS master manifest requests, `/v1/manifest` for HLS media manifest requests, and `/v1/dash` for DASH manifest requests.

Make sure that your CDN forwards all query parameters to AWS Elemental MediaTailor. MediaTailor relies on the query parameters to fulfill your VAST requests for personalized ads.

For server-side reporting, referencing a CDN in front of `/v1/segment` in ad segment requests helps prevent AWS Elemental MediaTailor from sending duplicate ad tracking beacons. When a player makes

a request for a `/v1/segment` ad, MediaTailor issues a 301 redirect to the actual `*.ts` segment. When MediaTailor sees that `/v1/segment` request, it issues a beacon call to track the view percentage of the ad. If the same player makes multiple requests for the same `/v1/segment` in one session, and your ADS can't de-duplicate requests, then MediaTailor issues multiple requests for the same beacon. Using a CDN to cache these 301 responses ensures that MediaTailor doesn't make duplicate beacon calls for repeated requests. For this path, you can use a high or default cache because cache-keys for these segments are unique.

To take advantage of these benefits, create behaviors in the CDN that route requests to the AWS Elemental MediaTailor configuration endpoint. Base the behaviors that you create on rules that differentiate requests for master HLS manifests, HLS manifests, DASH manifests, and reporting.

Requests follow these formats:

- HLS master manifest format

```
https://<playback-endpoint>/v1/master/<hashed-account-id>/<origin-id>/<master>.m3u8
```

  Example

```
https://a57b77e98569478b83c10881a22b7a24.mediatailor.us-east-1.amazonaws.com/v1/master/
a1bc06b59e9a570b3b6b886a763d15814a86f0bb/Demo/assetId.m3u8
```

- HLS manifest format

```
https://<playback-endpoint>/v1/manifest/<hashed-account-id>/<session-id>/
<manifestNumber>.m3u8
```

  Example

```
https://a57b77e98569478b83c10881a22b7a24.mediatailor.us-east-1.amazonaws.com/v1/manifest/
a1bc06b59e9a570b3b6b886a763d15814a86f0bb/c240ea66-9b07-4770-8ef9-7d16d916b407/0.m3u8
```

- DASH manifest format

```
https://<playback-endpoint>/v1/dash/<hashed-account-id>/<origin-id>/<assetName>.mpd
```

  Example

```
https://a57b77e98569478b83c10881a22b7a24.mediatailor.us-east-1.amazonaws.com/v1/dash/
a1bc06b59e9a570b3b6b886a763d15814a86f0bb/Demo/0.mpd
```

- Format for ad reporting request for server-side reporting

```
https://<playback-endpoint>/v1/segment/<origin-id>/<session-id>/<manifestNumber>/
<HLSSequenceNum>
```

  Example

```
https://a57b77e98569478b83c10881a22b7a24.mediatailor.us-east-1.amazonaws.com/v1/segment/
Demo/240ea66-9b07-4770-8ef9-7d16d916b407/0/440384
```

In the CDN, create a behavior that routes manifest requests to the AWS Elemental MediaTailor configuration endpoint. Base the behavior on a rule that includes a phrase to differentiate the manifest request from segment requests.

**Example Routing**

- Player requests to `https://CDN_Hostname/some/path/asset.m3u8` are routed to the AWS Elemental MediaTailor path `https://mediatailor.us-west-2.amazonaws.com/v1/session/configuration/endpoint` based on the keyword `*.m3u8` in the request.
- Player requests to `https://CDN_Hostname/some/path/asset.mpd` are routed to the AWS Elemental MediaTailor path `https://mediatailor.us-west-2.amazonaws.com/v1/dash/configuration/endpoint` based on the keyword `*.mpd` in the request.

# How AWS Elemental MediaTailor Handles BaseURLs for DASH

With server-side ad insertion, the content segments and ad segments come from different locations. In your DASH manifests, AWS Elemental MediaTailor manages URL settings based on your CDN configuration and the URLs specified in the manifest. MediaTailor uses the rules in the following list to manage the `BaseURL` settings in your DASH manifests for your content segments and ad segments.

AWS Elemental MediaTailor behavior for content segments:

- If you specify a **CDN content segment prefix** in your configuration, then MediaTailor makes sure that there is exactly one `BaseURL`, with your specified prefix, defined at the `MPD` level.
- If you do not specify a **CDN content segment prefix**, then MediaTailor uses the origin template manifest as follows:
  - If the origin template manifest contains one or more `BaseURL` settings at the `MPD` level, MediaTailor leaves them unmodified.
  - If the origin template manifest does not contain any `BaseURL` settings at the `MPD` level, MediaTailor adds one that is based on the origin `MPD` URL.

For ad segments, AWS Elemental MediaTailor does the following:

- If you specify a **CDN ad segment prefix** in your configuration, then MediaTailor ensures that each ad period has exactly one `BaseURL` setting, populated with the configured prefix.
- If you do not specify a **CDN ad segment prefix**, then MediaTailor adds exactly one `BaseURL` setting to each ad period that points to the ad content server that is set up by MediaTailor for serving ad segments.

# VAST in AWS Elemental MediaTailor

This topic covers the use of the following:

- Interactive Advertising Bureau (IAB)
- Video Ad Serving Template (VAST)
- Video Player Ad-Serving Interface Definition (VPAID)
- Video Multiple Ad Playlist (VMAP)

AWS Elemental MediaTailor supports VAST 3.0 and 2.0 and VMAP 1.0 for server-side ad insertion. AWS Elemental MediaTailor also supports the proxying of VPAID metadata through our client-side reporting API, for client-side ad insertion. For information about client-side reporting, see Client-side Reporting (p. 71).

For IAB specifications, see the following:

- VAST 3.0 – https://www.iab.com/guidelines/digital-video-ad-serving-template-vast-3-0/
- VMAP 1.0 – https://www.iab.com/guidelines/digital-video-multiple-ad-playlist-vmap-1-0-1/
- VPAID – https://www.iab.com/guidelines/digital-video-player-ad-interface-definition-vpaid-2-0/

**Topics**



# VAST Integration

To integrate your ad server with AWS Elemental MediaTailor, your ad server must send XML that conforms to the IAB specifications for the supported versions of VAST and VMAP. You can use a public VAST validator to ensure that your tags are well-formed.

Your ad server's VAST response must contain IAB compliant `TrackingEvents` elements and standard event types, like `impression`. If you don't include standard tracking events, AWS Elemental MediaTailor rejects the VAST response and doesn't provide an ad for the avail.

VAST 3.0 introduced support for ad pods, which is the delivery of a set of sequential linear ads. If a specific ad in an ad pod is not available, AWS Elemental MediaTailor logs an error on CloudWatch, in the interactions log of the ADS. It then tries to insert the next ad in the pod. In this way, MediaTailor iterates through the ads in the pod until it finds one that it can use.

## Targeting

To target specific players for your ads, you can create templates for your ad tags and URLs. For more information, see Dynamic Ad Variables in AWS Elemental MediaTailor (p. 60).

AWS Elemental MediaTailor proxies the player's `user-agent` and `x-forwarded-for` headers when it sends the ad server VAST request and when it makes the server-side tracking calls. Make sure that your ad server can handle these headers. Alternatively, you can use `[session.user_agent]` or `[session.client_ip]` and pass these values in query strings on the ad tag and ad URL. For more information, see Session Data (p. 62).

## Ad Calls

AWS Elemental MediaTailor calls your VAST ads URL as defined in your configuration. It substitutes any player-specific or session-specific parameters when making the ad call. MediaTailor follows up to three levels of VAST wrappers and redirects in the VAST response. In live streaming scenarios, MediaTailor makes ad calls simultaneously at the ad avail start for connected players. In practice, due to jitter, these ad calls can be spread out over a few seconds. Make sure that your ad server can handle the number of concurrent connections this type of calling requires. MediaTailor doesn't currently support pre-fetching VAST responses.

## Creative Handling

When AWS Elemental MediaTailor receives the ADS VAST response, for each creative it identifies the highest bitrate `MediaFile` for transcoding and uses this as its source. It sends this file to the on-the-fly transcoder for transformation into renditions that fit the player's master manifest bitrates and resolutions. For best results, make sure that your highest bitrate media file is a high-quality MP4 asset with valid manifest presets. When manifest presets aren't valid, the transcode jobs fail, resulting in no ad shown. Examples of presets that aren't valid include unsupported input file formats, like ProRes, and certain rendition specifications, like the resolution 855X481.

**Creative Indexing**

AWS Elemental MediaTailor uniquely indexes each creative by the value of the `id` attribute provided in the `<Creative>` element. If a creative's ID is not specified, MediaTailor uses the media file URL for the index.

The following example declaration shows the creative ID.

```
<Creatives>
    <Creative id="57859154776" sequence="1">
```

If you define your own creative IDs, use a new, unique ID for each creative. Don't reuse creative IDs. AWS Elemental MediaTailor stores creative content for repeated use, and finds each by its indexed ID. When a new creative comes in, the service first checks its ID against the index. If the ID is present, MediaTailor uses the stored content, rather than reprocessing the incoming content. If you reuse a creative ID, MediaTailor uses the older, stored ad and doesn't play your new ad.

# VPAID Handling

AWS Elemental MediaTailor supports VPAID for HLS. VPAID is not currently supported for DASH.

VPAID allows publishers to serve highly interactive video ads and to provide viewability metrics on their monetized streams. For information about VPAID, see the specification at https://www.iab.com/guidelines/digital-video-player-ad-interface-definition-vpaid-2-0/. AWS Elemental MediaTailor supports VPAID for HLS.

AWS Elemental MediaTailor supports a mix of server-side-stitched VAST MP4 linear ads and client-side-inserted VPAID interactive creatives in the same ad avail. It preserves the order in which they appear in the VAST response. MediaTailor follows VPAID redirects through a maximum of three levels of wrappers. The client-side reporting response includes the unwrapped VPAID metadata.

To use VPAID, follow these guidelines:

- Configure an MP4 slate for your VPAID creatives. AWS Elemental MediaTailor fills the VPAID ad slots with your configured slate, and provides VPAID ad metadata for the client player to use to

run the interactive ads. If you don't have a slate configured, when a VPAID ad appears, MediaTailor provides the ad metadata through client-side reporting as usual. It also logs an error in CloudWatch about the missing slate. For more information, see Slate Management (p. 69) and Creating a Configuration (p. 49).

- Use client-side reporting. AWS Elemental MediaTailor supports VPAID through our client-side reporting API. For more information, see Client-side Reporting (p. 71).

  It is theoretically possible to use the default server-side reporting mode with VPAID. However, if you use server-side reporting, you lose any information about the presence of the VPAID ad and the metadata surrounding it, because that is available only through the client-side API.
- In live scenarios, make sure that your ad avails, denoted by `EXT-X-CUE-OUT: Duration`, are large enough to accommodate any user interactivity on VPAID. For example, if the VAST XML specifies a VPAID ad that is 30 seconds long, implement your ad avail to be more than 30 seconds, to accommodate the ad. If you don't do this, you lose the VPAID metadata, because the remaining duration in the ad avail is not long enough to accommodate the VPAID ad.

# Dynamic Ad Variables in AWS Elemental MediaTailor

The AWS Elemental MediaTailor request to the ad decision server (ADS) includes information about the current viewing session, which helps the ADS choose the best ads to provide in its response. When you configure your ADS request, you specify the query parameters to use to convey the information.

The query parameters take the following forms:

- **Static values** – values that don't change from one session to the next. For example, the response type that MediaTailor expects from the ADS.
- **Session data** – dynamic values that are provided by MediaTailor for each session, for example, the session ID. For details, see Session Data (p. 62).
- **Player data** – dynamic values that are provided by the player for each session. These describe the content viewer and help the ADS to determine which ads MediaTailor should stitch into the stream. For details, see Player Data (p. 63).

## Passing Parameters to the ADS

**To pass session and player information to the ADS**

1. Work with the ADS to determine the information that it needs so that it can respond to an ad query from AWS Elemental MediaTailor.
2. Create a configuration in MediaTailor that uses a template ADS request URL that satisfies the ADS requirements. In the URL, include static parameters and include placeholders for dynamic parameters. Enter your template URL in the configuration's **Ad decision server** field.

   In the following example template URL, `correlation` provides session data, and `deviceType` provides player data:

   ```
   https://my.ads.server.com/path?
   correlation=[session.id]&deviceType=[player_params.deviceType]
   ```

3. On the player, configure the session initiation request for AWS Elemental MediaTailor to provide parameters for the player data. Include your parameters in the session initiation request, and omit them from subsequent requests for the session.

   The type of call that the player makes to initialize the session determines whether the player (client) or MediaTailor (server) provides ad-tracking reporting for the session. For information about these two options, see Ad Tracking Reporting in AWS Elemental MediaTailor (p. 70).

   Make one of the following types of calls, depending on whether you want server- or client-side ad-tracking reporting. In both of the example calls, `userID` is intended for the ADS and `auth_token` is intended for the origin:

   - (Option) Call for server-side ad-tracking reporting – Prefix the parameters that you want MediaTailor to send to the ADS with `ads`. Leave the prefix off for parameters that you want MediaTailor to send to the origin server:

The following examples show incoming requests for HLS and DASH to AWS Elemental MediaTailor. MediaTailor uses the `deviceType` in its request to the ADS and the `auth_token` in its request to the origin server.

HLS example:

```
GET master.m3u8?ads.deviceType=ipad&auth_token=kjhdsaf7gh
```

DASH example:

```
GET manifest.mpd?ads.deviceType=ipad&auth_token=kjhdsaf7gh
```

- (Option) Call for client-side ad-tracking reporting – Provide parameters for the ADS inside an `adsParams` object. Provide parameters that you want MediaTailor to send to the origin server as top-level objects.

  HLS example:

```
POST master.m3u8
    {
        "adsParams": {
           "deviceType": "ipad"
       }
       "auth_token": "kjhdsaf7gh"
    }
```

  DASH example:

```
POST manifest.mpd
    {
        "adsParams": {
           "deviceType": "ipad"
       }
       "auth_token": "kjhdsaf7gh"
    }
```

When the player initiates a session, AWS Elemental MediaTailor replaces the variables in the template ADS request URL with the session data and the player's `ads` parameters. It passes the remaining parameters from the player to the origin server.

The following examples show the calls to the ADS and origin server from AWS Elemental MediaTailor that correspond to the preceding player's session initialization call examples:

- MediaTailor calls the ADS with session data and the player's device type:

```
https://my.ads.server.com/path?correlation=896976764&deviceType=ipad
```

- MediaTailor calls the origin server with the player's authorization token.
  - HLS example:

```
https://my.origin.server.com/master.m3u8?auth_token=kjhdsaf7gh
```

  - DASH example:

```
https://my.origin.server.com/manifest.mpd?auth_token=kjhdsaf7gh
```

The following sections provide details for configuring session and player data.

# Session Data

To configure AWS Elemental MediaTailor to send session data to the ADS, in the template ADS URL, specify one or more of the variables listed in this section. You can use individual variables, and you can concatenate multiple variables to create a single value. MediaTailor generates some values and obtains the rest from sources like the manifest and the player's session initialization request.

You can use the following session data variables in your template ADS request URL configuration:

- **`[avail.random]`** – a random number between 0 and 10,000,000,000 that MediaTailor generates for each request to the ADS. Some ad servers use this parameter to enable features such as separating ads from competing companies.
- **`[scte.avail_num]`** – the value parsed by MediaTailor from the SCTE-35 field `avail_num`. MediaTailor can use this value to designate linear ad avail numbers.
- **`[scte.event_id]`** – the value parsed by MediaTailor from the SCTE-35 field `splice_event_id`, as a long number. MediaTailor uses this value to designate linear ad avail numbers or to populate ad server query strings, like ad pod positions.
- **`[scte.unique_program_id]`** – the value parsed by MediaTailor from the SCTE-35 `splice_insert` field `unique_program_id`. The ADS uses the unique program ID (UPID) to provide program-level ad targeting for live linear streams. If the SCTE-35 command is not splice insert, MediaTailor sets this to an empty value.
- **`[session.avail_duration_ms]`** – the duration in milliseconds of the ad availability slot. The default value is 300,000 ms. AWS Elemental MediaTailor obtains the duration value from the input manifest as follows:
  - For HLS, MediaTailor obtains the duration from the `#EXT-X-CUE-OUT: DURATION` or from values in the `#EXT-X-DATERANGE` tag. If the input manifest has a null, invalid, or 0 duration for the ad avail in those tags, MediaTailor uses the default.
  - For DASH, MediaTailor obtains the duration value from the event duration, if one is specified. Otherwise, it uses the default value.
- **`[session.avail_duration_secs]`** – the duration in seconds of the ad availability slot, or ad avail. MediaTailor determines this value the same way it determines `[session.avail_duration_ms]`.
- **`[session.client_ip]`** – the remote IP address that the MediaTailor request came from. If the `X-forwarded-for` header is set, then that value is what MediaTailor uses for the `client_ip`.
- **`[session.id]`** – a unique numeric identifier for the current playback session. All requests that a player makes for a session have the same id, so it can be used for ADS fields that are intended to correlate requests for a single viewing.
- **`[session.referer]`** – usually, the URL of the page that is hosting the video player. MediaTailor sets this variable to the value of the `Referer` header that the player used in its request to MediaTailor. If the player doesn't provide this header, MediaTailor leaves the **`[session.referer]`** empty. If you use a CDN or proxy in front of the manifest endpoint and you want this variable to appear, proxy the correct header from the player here.
- **`[session.user_agent]`** – the `User-Agent` header that MediaTailor received from the player's session initialization request. If you're using a CDN or proxy in front of the manifest endpoint, you must proxy the correct header from the player here.
- **`[session.uuid]`** – alternative to **`[session.id]`**. This is a unique identifier for the current playback session, such as the following:

```
e039fd39-09f0-46b2-aca9-9871cc116cde
```

### Example Examples

If the ADS requires a query parameter named `deviceSession` to be passed with the unique session identifier, the template ADS URL in AWS Elemental MediaTailor could look like the following:

```
https://my.ads.server.com/path?deviceSession=[session.id]
```

AWS Elemental MediaTailor automatically generates a unique identifier for each stream, and enters the identifier in place of `session.id`. If the identifier is `1234567`, the final request that MediaTailor makes to the ADS would look something like this:

```
https://my.ads.server.com/path?deviceSession=1234567
```

# Player Data

To configure AWS Elemental MediaTailor to send data received from the player to the ADS, in the template ADS URL, specify `player_params.<query_parameter_name>` variables. For example, if the player sends a query parameter named `user_id` in its request to MediaTailor, to pass that data in the ADS request, include `[player_params.user_id]` in the ADS URL configuration.

This allows you to control the query parameters that are included in the ADS request. Typically, you add a special query parameter that the ADS recognizes to the ADS request URL and provide key-value pairs as the value of the parameter.

The examples used in the following procedure use the following key-value pairs:

- *param1* with a value of *value1:*
- *param2* with a value of *value2:*

### To add query parameters as key-value pairs

1. In AWS Elemental MediaTailor, configure the ADS request template URL to reference the parameters. The following URL shows the inclusion of the example parameters:

   ```
   https://my.ads.com/path?param1=[player_params.param1]&param2=[player_params.param2]
   ```

2. (Optional) For server-side ad-tracking reporting, URL-encode the key-value pairs on the player. When MediaTailor receives the session initialization request, it URL-decodes the values once before substituting them into the ADS request URL.

   **Note**
   If your ADS requires a URL-encoded value, URL-encode the value twice on the player. This way, the decoding done by MediaTailor results in a once-encoded value for the ADS.

   For example, if the decoded representation of the values sent to the ADS is `param1=value1:&param2=value2:`, then the URL-encoded representation is `param1=value1%3A&param2=value2%3A`.

3. In the session initialization call from the player, pass the key-value pairs to MediaTailor as the value of a single query parameter. The following example calls provide the example key-value pairs for server- and client-side ad tracking reporting.
   - Example requests for server-side ad-tracking reporting - using URL-encoded pairs

     HLS:

```
<master>.m3u8?ads.param1=value1%3A&ads.param2=value2%3A
```

DASH:

```
<manifest>.mpd?ads.param1=value1%3A&ads.param2=value2%3A
```

- Example request for client-side ad-tracking reporting - with no URL-encoding

  HLS:

```
POST <master>.m3u8
    {
        "adsParams": {
           "param1": "value1:",
           "param2": "value2:"
       }
    }
```

  DASH:

```
POST <manifest>.mpd
    {
        "adsParams": {
           "param1": "value1:",
           "param2": "value2:"
       }
    }
```

For server-side reporting, MediaTailor decodes the parameters when the player request is received. For client-side reporting, it doesn't alter the parameters received in the JSON payload. MediaTailor sends the following request to the ADS:

```
https://my.ads.com/<path>?param1=value1:&param2=value2:
```

In this way, the `param1` and `param2` key-value pairs are included as first-class query parameters in the ADS request.

# Advanced Usage

You can customize the ADS request in many ways with player and session data. The only requirement is to include the ADS hostname.

The following examples show some of the ways that you can customize your request:

- Concatenate player parameters and session parameters to create new parameters. Example:

```
https://my.ads.com?key1=[player_params.value1][session.id]
```

- Use a player parameter as part of a path element. Example:

```
https://my.ads.com/[player_params.path]?key=value
```

- Use player parameters to pass both path elements and the keys themselves, rather than just values. Example:

```
https://my.ads.com/[player_params.path]?[player_params.key1]=[player_params.value1]
```

# Ad Behavior in AWS Elemental MediaTailor

This section covers how AWS Elemental MediaTailor manipulates the manifest to include the URLs for ads.

AWS Elemental MediaTailor replaces or inserts ads, depending on how the origin server configures the ad avails and on whether the content is VOD or live.

- With ad replacement, MediaTailor replaces content segments with ads.
- With ad insertion, MediaTailor inserts ad content where segments don't exist.

AWS Elemental MediaTailor also uses configured slates to fill gaps in ads and to manage VPAID ad handling.

**Topics**



# VOD Content Ad Behavior

AWS Elemental MediaTailor handles ads for VOD content for HLS. MediaTailor inserts or replaces ads in VOD streams based on how the origin server configured the `CUE-OUT/CUE-IN` (or `SCTE-OUT/SCTE-IN`) markers in the master manifest, or whether the ADS sends VMAP responses.

For ad behavior by marker configuration, see the following sections.

## No Cue Out or Cue In Markers

Although `CUE-OUT/IN` (or `SCTE-OUT/IN`) markers are the preferred way of signaling ad avails in a live manifest, the markers are not required for VOD content. If the manifest doesn't contain ad markers, MediaTailor makes a single call to the ADS and creates ad avails based on the response:

- If the ADS sends a VAST response, then MediaTailor inserts all ads from the response in an ad avail at the start of the manifest. This is a pre-roll.
- If the ADS sends a VMAP response, then MediaTailor uses the ad avail time offsets to create avails and insert them throughout the manifest at the specified times (pre-roll, mid-roll, or post-roll). MediaTailor uses all ads from each ad avail in the VMAP response for each ad avail in the manifest.

  **Note**
  When a segment overlaps an insertion point with VMAP for VOD content, MediaTailor rounds down to the nearest insertion point.

  **Tip**
  If you want to create mid-roll avails but your ADS doesn't support VMAP, then ensure that there are `CUE-OUT` (or `SCTE-OUT`) markers in the manifest. MediaTailor inserts ads at the markers, as described in the following sections.

# Cue Out and Cue In Markers Are Present

CUE-OUT/IN (or SCTE-OUT/IN) markers allow AWS Elemental MediaTailor to insert ads throughout the manifest. If the manifest contains markers, and the CUE-IN marker immediately follows the CUE-OUT marker (there are no segments between them), this informs MediaTailor that it is an ad insertion request.

The CUE-OUT markers should have no duration (or a duration of 0) specified, such as #EXT-X-CUE-OUT:0.

For post-rolls, CUE-OUT/IN markers must precede the last content segment. This is because the HLS spec requires tag decorators to be explicitly declared before a segment.

For example, consider the following declaration.

```
#EXT-X-CUE-OUT: 0
#EXT-X-CUE-IN
#EXTINF:4.000,
Videocontent.ts
#EXT-X-ENDLIST
```

AWS Elemental MediaTailor inserts a post-roll like the following.

```
#EXTINF:4.000,
Videocontent.ts
#EXT-X-DISCONTINUITY
#EXTINF:3.0,
Adsegment1.ts
#EXTINF:3.0,
Adsegment2.ts
#EXTINF:1.0,
Adsegment3.ts
#EXT-X-ENDLIST
```

You can't use multiple CUE-OUT/IN tags in succession to mimic ad pod behavior. This is because CUE-OUT/IN tags must be explicitly attached to a segment.

For example, the following declaration is invalid.

```
#EXT-X-CUE-OUT: 0
#EXT-X-CUE-IN
#EXT-X-CUE-OUT: 0
#EXT-X-CUE-IN
#EXT-X-CUE-OUT: 0
#EXT-X-CUE-IN
#EXTINF:4.000,
Videocontent.ts
```

The following declaration is valid.

```
#EXT-X-CUE-OUT: 0
#EXT-X-CUE-IN
#EXTINF:4.000,
Somecontent1.ts
#EXT-X-CUE-OUT: 0
#EXT-X-CUE-IN
#EXTINF:4.000,
Somecontent2.ts
#EXT-X-CUE-OUT: 0
```

```
#EXT-X-CUE-IN
#EXTINF:4.000,
Videocontent.ts
```

The preceding declaration results in an output like the following.

```
Ad 1
Somecontent.ts
Ad 2
Somecontent2.ts
Videocontent.ts
Post-Roll Ad 3
```

# Live Content Ad Behavior

In live streams, AWS Elemental MediaTailor always performs ad replacement, preserving the total time between the `CUE-OUT` and `CUE-IN` ad markers as closely as possible. The `CUE-OUT` can optionally include the `DURATION` attribute, which indicates the duration of the ad avail. Every `CUE-OUT` indicator must have a matching `CUE-IN` indicator in live workflows.

MediaTailor performs ad replacement for HLS and DASH live content. For information on how MediaTailor calculates ad avail placement and timing, see the section called “Ad Markers” (p. 18) and the section called “Ad Markers” (p. 22).

## Ad Selection and Replacement

AWS Elemental MediaTailor includes ads from the ADS VAST response as follows:

- If a duration is specified, MediaTailor selects a set of ads that fit into the duration and includes them.
- If no duration is specified, MediaTailor plays as many ads as it can until it encounters a cue-in indicator.

AWS Elemental MediaTailor adheres to the following guidelines during live ad replacement:

- MediaTailor tries to play complete ads, without clipping or truncation.
- Whenever MediaTailor encounters a cue-in indicator, it returns to the underlying content. This can mean truncating an ad that is currently playing.
- At the end of the duration, MediaTailor returns to the underlying content.
- If MediaTailor runs out of ads to play for the duration indicated, it plays the slate, if one is configured, or it returns to the underlying content. This happens most commonly when the ads that are available don't completely fill up the duration. This can also happen when the ADS response doesn't provide enough ads to fill the ad avail.

## Examples

- If the ad avail has a duration set to 70 seconds and the ADS response contains two 40-second ads, AWS Elemental MediaTailor plays one of the 40-second ads. In the time left over, it switches to the configured slate or underlying content. At any point during this process, if MediaTailor encounters a cue-in indicator, it cuts immediately to the underlying content.
- If the ad avail has a duration set to 30 seconds and the shortest ad provided by the ADS response is 40 seconds, MediaTailor plays no ads. If an ad slate is configured, MediaTailor plays that for 30 seconds or until it encounters a cue-in indicator. Otherwise, MediaTailor plays the underlying content.

# Slate Management

In AWS Elemental MediaTailor, you can configure a URL to an MP4 slate, to be used to fill gaps in media content. MediaTailor inserts the slate into unfilled and partially filled ad avails. MediaTailor downloads the slate from the MP4 URL and transcodes it to the same renditions as your content, for smooth transitions between the two. The slate may be played in a loop if the duration of the remaining ad avail allows for it.

Configuring a slate is optional in all situations except where VPAID is used:

- For non-VPAID situations, if you don't configure a slate, MediaTailor handles unfilled and partially filled ad avails by showing the underlying stream content.
- For VPAID, you must configure a slate. MediaTailor inserts the slate for the duration of the VPAID ad. In certain cases, to accommodate user interactivity, this duration might be slightly higher than the duration of the VPAID ad as reported by VAST. The video player then handles the VPAID ad based on the client-side reporting metadata that MediaTailor returns. For information about client-side reporting, see the section called “Client-side Reporting” (p. 71). For information about VPAID, see the section called “VPAID” (p. 58).

The slate that you configure must be a high-quality MP4 asset that contains both audio and video. Empty audio slates sometimes cause playback issues on some players.

AWS Elemental MediaTailor shows the slate for the following situations:

- To fill in time that's not fully used by an ad replacement
- If the ADS responds with a blank VAST or VMAP response
- For error conditions, such as ADS timeout
- If ads are longer than the live ad avail window
- If an ad isn't available

AWS Elemental MediaTailor always shows the slate near the end of the ad avail.

# Ad Tracking Reporting in AWS Elemental MediaTailor

Beacons are sent to the ad server to track and report on how much of an ad that a viewer has watched. AWS Elemental MediaTailor provides server-side ad reporting (MediaTailor tracks the ad and sends beacons) or client-side tracking (the client player tracks the ad and sends beacons). The type of reporting that is used in a playback session depends on the request that the player uses to initiate the session in MediaTailor.

**Topics**



## Server-side Reporting

AWS Elemental MediaTailor defaults to server-side reporting. With server-side reporting, when the player requests an ad URL from the manifest, the service reports ad consumption directly to the ad tracking URL. After the player initializes a playback session with MediaTailor, no further input is required from you or the player to perform server-side reporting. As each ad is played back, MediaTailor sends beacons to the ad server to report how much of the ad has been viewed. MediaTailor sends beacons for the start of the ad and for the ad progression in quartiles: the first quartile, midpoint, third quartile, and ad completion.

**To perform server-side ad reporting**

- From the player, initialize a new MediaTailor playback session using a request in one of the following formats, according to your protocol:
  - Example: HLS format

    ```
    GET <mediatailorURL>/v1/master/<hashed-account-id>/<origin-id>/<asset-id>?ads.<key-value-pairs-for-ads>&<key-value-pairs-for-origin-server>
    ```

  - Example: DASH format

    ```
    GET <mediatailorURL>/v1/dash/<hashed-account-id>/<origin-id>/<asset-id>?ads.<key-value-pairs-for-ads>&<key-value-pairs-for-origin-server>
    ```

  The key-value pairs are the dynamic targeting parameters for ad tracking. For information about adding parameters to the request, see *Dynamic Ad Variables* (p. 60).

AWS Elemental MediaTailor responds to the request with the manifest URL. The manifest contains URLs for the media manifests. The media manifests contain embedded links for ad segment requests.

When the player requests playback from an ad segment URL (`/v1/segment` path), AWS Elemental MediaTailor sends the appropriate beacon to the ad server through the ad tracking URLs. At the same time, the service issues a redirect to the actual `*.ts` ad segment. The ad segment is either in the Amazon

CloudFront distribution where MediaTailor stores transcoded ads, or in the content distribution network (CDN) where you have cached the ad.

# Client-side Reporting

With client-side reporting, AWS Elemental MediaTailor proxies the ad tracking URL to the client player. The player then performs all ad-tracking activities.

Client-side reporting enables functionality like the following:

- Trick play for VOD, where players display visual feedback during fast forward and rewind.
- Advanced playback behaviors that require player development, like no skip-forward and countdown timers on ad avails.

Use client-side reporting for VPAID functionality. For more information, see VPAID Handling (p. 58). The client-side reporting response includes additional metadata about the VPAID creative.

**To perform client-side ad reporting**

1. On the player, construct a JSON message body for the session initialization request to AWS Elemental MediaTailor:

   - Provide parameters that MediaTailor should pass to the ADS inside an `adsParams` object. These parameters correspond to `[player_params.param]` settings in the ADS template URL of the MediaTailor configuration.
   - Provide parameters that you want MediaTailor to send to the origin server as top-level objects.

   Example: HLS

   ```
   POST master.m3u8
       {
           "adsParams": {
              "deviceType": "ipad"
           }
           "auth_token": "kjhdsaf7gh"
       }
   ```

   Example: DASH

   ```
   POST manifest.mpd
       {
           "adsParams": {
              "deviceType": "ipad"
           }
           "auth_token": "kjhdsaf7gh"
       }
   ```

   For more information about providing dynamic ad variables, see *Dynamic Ad Variables* (p. 60).
2. From the player, use your constructed JSON to initialize a new MediaTailor playback session. Format your request like the following.

   ```
   POST <mediatailorURL>/v1/session/<hashed-account-id>/<origin-id>/<asset-id>
       {
           "adsParams": {
               "param1": "value1",
   ```

```
            "param2": "value2",
            "param3": "value3"
        }
        "originServerParam1": "originValue1",
        "originServerParam2": "originValue2"
    }
```

AWS Elemental MediaTailor responds to the request with two relative URLs, one for the manifest and one for the tracking endpoint:

- Manifest – used to retrieve content manifests and ad segments

  Example: HLS

  ```
  /v1/master/<hashed-account-id>/<origin-id>/<asset-id>?aws.sessionId=<session>
  ```

  Example: DASH

  ```
  /v1/dash/<hashed-account-id>/<origin-id>/<asset-id>?aws.sessionId=<session>
  ```

- Tracking – used to poll for upcoming ad avails

  Example

  ```
  /v1/tracking/<hashed-account-id>/<origin-id>/<asset-id>/<session>
  ```

3. Construct the full manifest and tracking URLs by prefixing the relative URLs from MediaTailor with *<mediatailorURL>*.
4. Program the player to periodically poll the tracking URL and manage ad avails accordingly. When an ad is coming, the response from AWS Elemental MediaTailor to the player's polling request contains a JSON object that specifies the time offsets for the ad avails. The offsets are relative to when the player initiated the session. You can use them when programming specific behaviors in the player, such as preventing the viewer from skipping past the ads. The response also includes duration, timing, and identification information.

   The response from MediaTailor can include the following values:

   - `adId`: For HLS, the sequence number associated with the beginning of the ad. For DASH, the period ID of the ad.
   - `adParameters`: String of ad parameters from VAST VPAID, which AWS Elemental MediaTailor passes to the player.
   - `adSystem`: The name for the system that serves the ad.
   - `adTitle`: The title for the ad.
   - `apiFramework`: Set this to `VPAID` to tell the player that this is a VPAID ad.
   - `availId`: For HLS, the sequence number associated with the start of the ad avail. For DASH, the period ID of the ad avail, which is usually the period ID of the content that is to be replaced with an ad.
   - `beaconUrls`: Where to send each ad beacon.
   - `bitrate`: Bitrate of the video asset. This is not typically included for an executable asset.
   - `companionAds`: One or more companion ad content specifications, each of which specifies a resource file to use. Companion ads accompany the ad avail, and are used to provide content like a frame around the ad or a banner to display near the video.
   - `creativeId`: The `Id` attribute of the `Creative` tag for the ad.
   - `delivery`: This indicates the protocol used, and can be set to either `progressive` or `streaming`.

- `duration`: Length in ISO 8601 seconds format. The response includes durations for the entire ad avail and for each ad and beacon (though beacon durations are always zero). For VPAID Handling (p. 58), the duration conveyed is the MP4 slate duration. This duration typically is slightly larger than the XML duration conveyed in VAST due to transcoder and segment duration configurations. You can interpret this as the maximum amount of time that you have available to fill with a VPAID ad without incurring drift.
- `durationInSeconds`: Length in seconds format. The response includes durations for the entire ad avail and for each ad and beacon (though beacon durations are always zero).
- `eventId`: For HLS, the sequence number that is associated with the beacon. For DASH, the `ptsTime` of the start of the ad.
- `eventType`: Type of beacon.
- `height`: Height of the video asset.
- `maintainAspectRatio`: Indicates whether to maintain the aspect ratio while scaling.
- `mediaFilesList`: Specifies video and other assets that the player needs for the ad avail.
- `mediaFileUri`: URI that points to either an executable asset or video asset. Example. `https://myad.com/ad/ad134/vpaid.js`.
- `mediaType`: Typically either `JavaScript` or `Flash` for executable assets.
- `mezzanine`: The mezzanine MP4 asset, specified if the VPAID ad includes one. Example. `https://gcdn.2mdn.net/videoplayback/id/itag/ck2/file/file.mp4`.
- `scalable`: Indicates whether to scale the video to other dimensions.
- `startTime`: Time position in ISO 8601 seconds format, relative to the beginning of the playback session. The response includes start times for the entire ad avail and for each ad and beacon.
- `startTimeInSeconds`: Time position in seconds format, relative to the beginning of the playback session. The response includes start times for the entire ad avail and for each ad and beacon.
- `vastAdId`: The `Id` attribute of the `Ad` tag.
- `width`: Width of the video asset.

The following example responses indicate that an ad is coming.

```
{
  "avails": [
    {
      "ads": [
        {
          "adId": "8104385",
          "duration": "PT15.100000078S",
          "durationInSeconds": 15.1,
          "startTime": "PT17.817798612S",
          "startTimeInSeconds": 17.817,
          "trackingEvents": [
    {
              "beaconUrls": [
                "http://exampleadserver.com/tracking?event=impression"
              ],
              "duration": "PT15.100000078S",
              "durationInSeconds": 15.1,
              "eventId": "8104385",
              "eventType": "impression",
              "startTime": "PT17.817798612S",
              "startTimeInSeconds": 17.817
            },
            {
              "beaconUrls": [
                "http://exampleadserver.com/tracking?event=start"
              ],
```

```
            "duration": "PT0S",
            "durationInSeconds": 0.0,
            "eventId": "8104385",
            "eventType": "start",
            "startTime": "PT17.817798612S",
            "startTimeInSeconds": 17.817
          },
  {
            "beaconUrls": [
              "http://exampleadserver.com/tracking?event=firstQuartile"
            ],
            "duration": "PT0S",
            "durationInSeconds": 0.0,
            "eventId": "8104386",
            "eventType": "firstQuartile",
            "startTime": "PT21.592798631S",
            "startTimeInSeconds": 21.592
          },
   {
            "beaconUrls": [
              "http://exampleadserver.com/tracking?event=midpoint"
            ],
            "duration": "PT0S",
            "durationInSeconds": 0.0,
            "eventId": "8104387",
            "eventType": "midpoint",
            "startTime": "PT25.367798651S",
            "startTimeInSeconds": 25.367
          },
          {
            "beaconUrls": [
              "http://exampleadserver.com/tracking?event=thirdQuartile"
            ],
            "duration": "PT0S",
            "durationInSeconds": 0.0,
            "eventId": "8104388",
            "eventType": "thirdQuartile",
            "startTime": "PT29.14279867S",
            "startTimeInSeconds": 29.142
          },
          {
            "beaconUrls": [
              "http://exampleadserver.com/tracking?event=complete"
            ],
            "duration": "PT0S",
            "durationInSeconds": 0.0,
            "eventId": "8104390",
            "eventType": "complete",
            "startTime": "PT32.91779869S",
            "startTimeInSeconds": 32.917
          }
        ]
      }
    ],
    "availId": "8104385",
    "duration": "PT15.100000078S",
    "durationInSeconds": 15.1,
    "meta": null,
    "startTime": "PT17.817798612S",
    "startTimeInSeconds": 17.817
  }
 ]
}
```

```
{
  "avails": [
    {
      "ads": [
        {
          "adId": "6744037",
          "mediaFiles": {
            "mezzanine": "https://gcdn.2mdn.net/videoplayback/id/itag/ck2/file/
file.mp4",
            "mediaFilesList": [
              {
                "mediaFileUri": "https://myad.com/ad/ad134/vpaid.js",
                "delivery": "progressive",
                "width": 176,
                "height": 144,
                "mediaType": "application/javascript",
                "scalable": false,
                "maintainAspectRatio": false,
                "apiFramework": "VPAID"
              },
              {
                "mediaFileUri": "https://myad.com/ad/ad134/file.mp4",
                "delivery": "progressive",
                "width": 640,
                "height": 360,
                "mediaType": "video/mp4",
                "scalable": false,
                "maintainAspectRatio": false
              },
              ...
            ],
            "adParameters": "[{'ads':[{"url":"https://myads/html5/media/
LinearVPAIDCreative.mp4","mimetype":"video/mp4"]}]",
            "duration": "PT15.066667079S",
            "durationInSeconds": 15.066,
            "startTime": "PT39.700000165S",
            "startTimeInSeconds": 39.7,
            "trackingEvents": [
              {
                "beaconUrls": [
                  "https://beaconURL.com"
                ],
                "duration": "PT15.066667079S",
                "durationInSeconds": 15.066,
                "eventId": "6744037",
                "eventType": "impression",
                "startTime": "PT39.700000165S",
                "startTimeInSeconds": 39.7
              },
              ...
            ]
          }
        },
        ...
      ],
      "availId": "6744037",
      "duration": "PT45.166667157S",
      "durationInSeconds": 45.166,
      "meta": null,
      "startTime": "PT39.700000165S",
      "startTimeInSeconds": 39.7
    }
  ]
```

```
}
```

```
{
    "avails": [
      {
        "ads": [
          {
            "adId": "3348173",
            "adParameters": null,
            "duration": "PT12.700001178S",
            "durationInSeconds": 12.7,
            "mediaFiles": null,
            "startTime": "PT1M56.060003037S",
            "startTimeInSeconds": 116.06,
            "trackingEvents": [],
            "companionAds": [
              {
                   "sequence": "",
                   "attributes": {
                     "width": "REQUIRED",
                     "height": "REQUIRED",
                     "id": "",
                     "assetWidth": "",
                     "assetHeight": "",
                     "expandedWidth": "",
                     "expandedHeight": "",
                     "apiFramework": "",
                     "adSlotId": "",
                     "pxratio": "",
                     "renderingMode": ""
                   },
                   "staticResource": "",
                   "iFrameResource": "",
                   "htmlResource": "<![CDATA[<!doctype html><html><head><meta name=
\"viewport\" content=\"width=1, initial-scale=1.0, minimum-scale=1.     0, . . . ]]>",
                   "adParameters": "",
                   "altText": "",
                   "companionClickThrough": "",
                   "companionClickTracking": "",
                   "trackingEvents": [
                     {
                       "tracking": {}
                     }
                   ]
              }
            ]
          }
        ],
        "availId": "3348173",
        "duration": "PT12.700001178S",
        "durationInSeconds": 12.7,
        "meta": null,
        "startTime": "PT1M56.060003037S",
        "startTimeInSeconds": 116.06
      }
    ]
  }
```

# Tagging AWS Elemental MediaTailor Resources

A *tag* is a metadata label that you assign or that AWS assigns to an AWS resource. Each tag consists of a *key* and a *value*. For tags that you assign, you define the key and value. For example, you might define the key as `stage` and the value for one resource as `test`.

Tags help you do the following:

- Identify and organize your AWS resources. Many AWS services support tagging, so you can assign the same tag to resources from different services to indicate that the resources are related. For example, you could assign the same tag to an AWS Elemental MediaPackage channel and endpoint that you assign to an AWS Elemental MediaTailor configuration.
- Track your AWS costs. You activate these tags on the AWS Billing and Cost Management dashboard. AWS uses the tags to categorize your costs and deliver a monthly cost allocation report to you. For more information, see Use Cost Allocation Tags in the AWS Billing and Cost Management User Guide.
- Control access to your AWS resources. For more information, see Controlling Access Using Tags in the IAM User Guide.

For tips on using tags, see the AWS Tagging Strategies post on the *AWS Answers* blog.

The following sections provide more information about tags for AWS Elemental MediaTailor.

## Supported Resources in AWS Elemental MediaTailor

The following resource in AWS Elemental MediaTailor supports tagging:

- Configurations

## Tag Restrictions

The following basic restrictions apply to tags on AWS Elemental MediaTailor resources:

- Maximum number of tags that you can assign to a resource – 50
- Maximum key length – 128 Unicode characters
- Maximum value length – 256 Unicode characters
- Valid characters for key and value – a-z, A-Z, 0-9, space, and the following characters: _ . : / = + - and @
- Keys and values are case sensitive
- Don't use `aws:` as a prefix for keys; it's reserved for AWS use

# Managing Tags in AWS Elemental MediaTailor

You set tags as properties on a resource. You can add, edit, and delete tags through the AWS Elemental MediaTailor API or the AWS CLI. For more information, see the *AWS Elemental MediaTailor API Reference*.

# Playback Errors Returned by AWS Elemental MediaTailor

This section provides information about the HTTP error codes that you might receive while testing your player software and during the normal processing of player requests.

> **Note**
> You might also receive errors from the AWS Elemental MediaTailor API, during configuration operations like `PutPlaybackConfiguration` and `GetPlaybackConfiguration`. For information about those types of errors, see the AWS Elemental MediaTailor API Reference.

When your player sends a request to AWS Elemental MediaTailor, either directly or through a CDN, MediaTailor responds with a status code. If MediaTailor successfully handles the request, it returns the HTTP status code `200 OK`, indicating success, along with the populated manifest. If the request is unsuccessful, MediaTailor returns an HTTP status code, an exception name, and an error message.

AWS Elemental MediaTailor returns two classes of errors:

- **Client errors** – errors that are usually caused by a problem in the request itself, like an improperly formatted request, an invalid parameter, or a bad URL. These errors have an HTTP `4xx` response code.
- **Server errors** – errors that are usually caused by a problem with MediaTailor or one of its dependencies, like the ad decision server (ADS) or the origin server. These errors have an HTTP `5xx` response code.

**Topics**



## Client Playback Errors Returned by AWS Elemental MediaTailor

General guidance:

- You can find detailed information for most errors in the headers and body of the response.
- For some errors, you need to check your configuration settings. You can retrieve the settings for your playback configuration from AWS Elemental MediaTailor. For the API, the resource is `GetPlaybackConfiguration/Name`. For details, see the AWS Elemental MediaTailor API Reference.

The following table lists the client error codes that are returned by the manifest manipulation activities of AWS Elemental MediaTailor, probable causes, and actions you can take to resolve them.

| Co | Exception Name | Meaning | What To Do |
|---|---|---|---|
| 400 | `BadRequestException` | MediaTailor is unable to service the request due to one or more errors in | Check that your request is properly formatted and contains accurate |

| Co | Exception Name | Meaning | What To Do |
|---|---|---|---|
| | | formatting or content. A parameter might be improperly formatted, or the request might contain an invalid playback configuration or session ID. | information. Make sure that the playback endpoint setting on the player matches the `ManifestEndpointPrefix` setting returned by `GetPlaybackConfiguration`. Retry your request. |
| 403 | `AccessDeniedException` | The host header provided in the request doesn't match the manifest endpoint prefix that is configured in the MediaTailor playback URL. Your CDN might be misconfigured. | Check your CDN settings and make sure that you are using the correct manifest endpoint prefix for MediaTailor. Retry your request. |
| 404 | `NotFoundException` | MediaTailor is unable to find the information specified. Possible reasons include a URL that doesn't map to anything in the service, a configuration that isn't defined, or a session that is unavailable. | Check your configuration and the validity of your request, and then reinitialize the session. |
| 409 | `ConflictException` | A player tried to load multiple playlists simultaneously for a single session. As a result, MediaTailor detected a session consistency conflict. This problem occurs for HLS players. | Make sure that your player requests playlists one at a time. This is in accordance with the HLS specification. |
| 410 | `Gone` | An AWS Support operator has blocked a player session or customer configuration. AWS Support does this in rare circumstances when we detect a very high volume of 4xx requests coming from errant traffic for a single session or configuration. | If you think that the request shouldn't be blocked, contact AWS Support. They can look into it and remove the blocking filter, if appropriate. |

If you need further assistance, contact AWS Support.

# Server Playback Errors Returned by AWS Elemental MediaTailor

General guidance:

- You can find detailed information for most errors in the headers and body of the response.
- For some errors, you need to check your configuration settings. You can retrieve the settings for your playback configuration from AWS Elemental MediaTailor. For the API, the resource is `GetPlaybackConfiguration/`*`Name`*. For details, see the AWS Elemental MediaTailor API Reference.

The following table lists the server error codes returned by the manifest manipulation activities of AWS Elemental MediaTailor, probable causes, and actions you can take to resolve them.

| Co | Exception Name | Meaning | What To Do |
|---|---|---|---|
| 500 | `InternalServiceError` | Unhandled exception. | Retry the request. If the problem persists, check the reported health of MediaTailor for your AWS Region at https://status.aws.amazon.com/. |
| 502 | `BadGatewayException` | Either the origin server address or the ad decision server (ADS) address is invalid. Examples of invalid addresses are a private IP address and `localhost`. | Make sure that your configuration has the correct settings for your ADS and origin server, and then retry the request. |
| 502 | `UnsupportedManifestException` | Either the origin manifest has changed so that MediaTailor can't personalize it or MediaTailor doesn't support the origin's manifest format. | This might affect only the individual session. Reinitialize the session. You can usually accomplish this by refreshing the page in the viewer. If the problem persists, verify that MediaTailor supports the origin's manifest format. For information, see Manifest Handling (p. 17). |
| 503 | `LoadShed` | MediaTailor experienced a resource constraint while servicing your request. | Retry the request. If the problem persists, check the reported health of MediaTailor for your AWS Region at https://status.aws.amazon.com/. |
| 503 | `ThrottlingException` | Your transactions per second have reached your limit, and MediaTailor is throttling your use. | Retry the request. You can also check the reported health of MediaTailor for your AWS Region at https://status.aws.amazon.com/. You might want to increase the limit on your transactions per second. For more information, see the section called "Soft Limits" (p. 129). |
| 504 | `GatewayTimeoutException` | A timeout occurred while MediaTailor was contacting the origin server. | Retry the request. If the problem persists, check the health of the origin server and make sure the origin server is responding within the content origin server timeout that is listed at the section called "Hard Limits" (p. 129). |

If you need further assistance, contact AWS Support.

# Playback Error Examples

This section lists some examples of the playback errors that you might see in command line interactions with AWS Elemental MediaTailor.

The following example shows the result when a timeout occurs between AWS Elemental MediaTailor and either the ad decision server (ADS) or the origin server.

```
~[]> curl -vvv https://111122223333444455556666123456789012.mediatailor.us-
west-2.amazonaws.com/v1/master/123456789012/Multiperiod_DASH_Demo/index.mpd
*   Trying 54.186.133.224...
* Connected to 111122223333444455556666123456789012.mediatailor.us-west-2.amazonaws.com
 (11.222.333.444) port 555 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: mediatailor.us-west-2.amazonaws.com
* Server certificate: Amazon
* Server certificate: Amazon Root CA 1
* Server certificate: Starfield Services Root Certificate Authority - G2
> GET /v1/master/123456789012/Multiperiod_DASH_Demo/index.mpd HTTP/1.1
> Host: 111122223333444455556666123456789012.mediatailor.us-west-2.amazonaws.com
> User-Agent: curl/7.43.0
> Accept: */*
>
< HTTP/1.1 504 Gateway Timeout
< Date: Thu, 29 Nov 2018 18:43:14 GMT
< Content-Type: application/json
< Content-Length: 338
< Connection: keep-alive
< x-amzn-RequestId: 123456789012-123456789012
< x-amzn-ErrorType: GatewayTimeoutException:http://internal.amazon.com/coral/
com.amazon.elemental.midas.mms.coral/
<
* Connection #0 to host 111122223333444455556666123456789012.mediatailor.us-
west-2.amazonaws.com left intact
{"message":"failed to generate manifest: Unable to obtain template playlist. origin URL:
[https://777788889999.mediapackage.us-west-2.amazonaws.com/out/v1/444455556666111122223333/
index.mpd], asset path: [index.mpd], sessionId:[12345678901212345678901 2] customerId:
[123456789012]"}%
```

# Security in AWS Elemental MediaTailor

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS Elemental MediaTailor. The following topics show you how to configure MediaTailor to meet your security and compliance objectives. You also learn how to use other AWS services to monitor and secure your MediaTailor resources.

**Topics**



## Data Protection in AWS Elemental MediaTailor

AWS Elemental MediaTailor doesn't encrypt or decrypt data in its management of content manifests or in its communication with servers, CDNs, or players. MediaTailor doesn't require that you supply any customer data or other sensitive information.

Don't put sensitive information, like customer account numbers, credit card information, or passwords, into free-form fields or query parameters. This applies to all use of AWS Elemental MediaTailor, including the console, API, SDKs, and the AWS CLI. Any data that you enter into the service might get picked up for inclusion in diagnostic logs.

When you provide a URL to an external server, don't include unencrypted credentials information in the URL to validate your request to that server.

## Identity and Access Management for AWS Elemental MediaTailor

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and

*authorized* (have permissions) to use AWS Elemental MediaTailor resources. IAM is an AWS service that you can use with no additional charge.

This section provides background and additional information about the setup procedures you follow to use AWS Elemental MediaTailor. See Setting Up AWS Elemental MediaTailor (p. 5).

**Topics**



# Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in AWS Elemental MediaTailor.

**Service user** – If you use the AWS Elemental MediaTailor service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more MediaTailor features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in MediaTailor, see Troubleshooting AWS Elemental MediaTailor Identity and Access (p. 92).

**Service administrator** – If you're in charge of AWS Elemental MediaTailor resources at your company, you probably have full access to MediaTailor. It's your job to determine which MediaTailor features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with MediaTailor, see How AWS Elemental MediaTailor Works with IAM (p. 88).

**IAM administrator** – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS Elemental MediaTailor. To view example MediaTailor identity-based policies that you can use in IAM, see AWS Elemental MediaTailor Identity-Based Policy Examples (p. 89).

# Authenticating With Identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see The IAM Console and Sign-in Page in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication, or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email or your IAM user name. You can access AWS programmatically using your root user or IAM user access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol

for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.

## AWS Account Root User

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

## IAM Users and Groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing Access Keys for IAM Users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to Create an IAM User (Instead of a Role) in the *IAM User Guide*.

## IAM Roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM Roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access.

However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the *IAM User Guide*.

- **AWS service access** – A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances in the *IAM User Guide*.

To learn whether to use IAM roles, see When to Create an IAM Role (Instead of a User) in the *IAM User Guide*.

# Managing Access Using Policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an entity (root user, IAM user, or IAM role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON Policies in the *IAM User Guide*.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

## Identity-Based Policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, role, or group. These policies control what actions that identity can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM Policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing Between Managed Policies and Inline Policies in the *IAM User Guide*.

## Resource-Based Policies

Resource-based policies are JSON policy documents that you attach to a resource such as an Amazon S3 bucket. Service administrators can use these policies to define what actions a specified principal (account member, user, or role) can perform on that resource and under what conditions. Resource-based policies are inline policies. There are no managed resource-based policies. AWS Elemental MediaTailor doesn't support resource-based policies.

## Access Control Lists (ACLs)

Access control policies (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they are the only policy type that does not use the JSON policy document format. Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access Control List (ACL) Overview in the *Amazon Simple Storage Service Developer Guide*. AWS Elemental MediaTailor doesn't support ACLs.

## Other Policy Types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions Boundaries for IAM Entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs Work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session Policies in the *IAM User Guide*.

## Multiple Policy Types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy Evaluation Logic in the *IAM User Guide*.

# Learn More

For more information about identity and access management for AWS Elemental MediaTailor, see the following:

- How AWS Elemental MediaTailor Works with IAM (p. 88)
- Troubleshooting AWS Elemental MediaTailor Identity and Access (p. 92)

# How AWS Elemental MediaTailor Works with IAM

To get a high-level view of how AWS Elemental MediaTailor and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

**Topics**



## AWS Elemental MediaTailor Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. AWS Elemental MediaTailor supports specific actions, resources, and condition keys. To learn about all of the elements, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

### Actions

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in AWS Elemental MediaTailor prefix the action with `mediatailor:`. For example, to grant someone permission to run the MediaTailor `ListTagsForResource` API operation, you include the `mediatailor:ListTagsForResource` action in their policy. Policy statements must include either an `Action` or `NotAction` element. MediaTailor defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows.

```
"Action": [
      "mediatailor:action1",
      "mediatailor:action2"
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `List`, include the following action:

```
"mediatailor:List*"
```

For a list of AWS Elemental MediaTailor actions, see Actions Defined by AWS Elemental MediaTailor in the *IAM User Guide*.

### Resources

AWS Elemental MediaTailor doesn't support specifying resource ARNs in a policy.

### Condition Keys

AWS Elemental MediaTailor doesn't provide service-specific condition keys, but it does support using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

### Examples

To view examples of AWS Elemental MediaTailor identity-based policies, see AWS Elemental MediaTailor Identity-Based Policy Examples (p. 89).

## AWS Elemental MediaTailor Resource-Based Policies

AWS Elemental MediaTailor doesn't support resource-based policies.

## Authorization Based on AWS Elemental MediaTailor Tags

You can attach tags to AWS Elemental MediaTailor resources and pass tags in a request to MediaTailor. To control access using tags, you provide tag information in the condition element of a policy using the `mediatailor:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys. For more information about tagging MediaTailor resources, see Tagging AWS Elemental MediaTailor Resources (p. 77).

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see Viewing AWS Elemental MediaTailor Configurations Based on Tags (p. 92).

## AWS Elemental MediaTailor IAM Roles

An IAM role is an entity within your AWS account that has specific permissions.

### Using Temporary Credentials with AWS Elemental MediaTailor

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

AWS Elemental MediaTailor supports using temporary credentials.

### Service-Linked Roles

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

AWS Elemental MediaTailor doesn't support service-linked roles.

### Service Roles

This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

AWS Elemental MediaTailor doesn't support service roles.

# AWS Elemental MediaTailor Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify AWS Elemental MediaTailor resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An

IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating Policies on the JSON Tab in the *IAM User Guide*.

## Policy Best Practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete AWS Elemental MediaTailor resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get Started Using AWS Managed Policies** – To start using AWS Elemental MediaTailor quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get Started Using Permissions With AWS Managed Policies in the *IAM User Guide*.
- **Grant Least Privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant Least Privilege in the *IAM User Guide*.
- **Enable MFA for Sensitive Operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.
- **Use Policy Conditions for Extra Security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

## Using the AWS Elemental MediaTailor Console

To access the AWS Elemental MediaTailor console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the MediaTailor resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM users or roles) with that policy.

To ensure that those entities can still use the AWS Elemental MediaTailor console, also attach the following AWS managed policy to the entities. For more information, see Adding Permissions to a User in the *IAM User Guide*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": [
                "arn:aws:iam::*:user/${aws:username}"
            ]
        },
```

```
        {
            "Sid": "ListUsersViewGroupsAndPolicies",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

## Allow Users to View Their Own Permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "ViewOwnUserInfo",
               "Effect": "Allow",
               "Action": [
                   "iam:GetUserPolicy",
                   "iam:ListGroupsForUser",
                   "iam:ListAttachedUserPolicies",
                   "iam:ListUserPolicies",
                   "iam:GetUser"
               ],
               "Resource": [
                   "arn:aws:iam::*:user/${aws:username}"
               ]
           },
           {
               "Sid": "NavigateInConsole",
               "Effect": "Allow",
               "Action": [
                   "iam:GetGroupPolicy",
                   "iam:GetPolicyVersion",
                   "iam:GetPolicy",
                   "iam:ListAttachedGroupPolicies",
                   "iam:ListGroupPolicies",
                   "iam:ListPolicyVersions",
                   "iam:ListPolicies",
                   "iam:ListUsers"
               ],
               "Resource": "*"
           }
       ]
   }
```

## Viewing AWS Elemental MediaTailor Configurations Based on Tags

You can use conditions in your identity-based policy to control access to AWS Elemental MediaTailor resources based on tags. This example shows how you might create a policy that allows a user to view only their own configuration. Permission is granted only if the configuration tag `Owner` has the value of the user's user name. The policy also grants the permissions necessary to complete the view action through the console.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListConfigurationsInConsole",
            "Effect": "Allow",
            "Action": "mediatailor:ListPlaybackConfigurations",
            "Resource": "*"
        },
        {
            "Sid": "ViewConfigurationIfOwner",
            "Effect": "Allow",
            "Action": "mediatailor:GetPlaybackConfiguration",
            "Resource": "arn:aws:mediatailor:*:*:configuration/*",
            "Condition": {
                "StringEquals": {"mediatailor:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

You can attach this policy to the IAM users in your account. For a user named `richard-roe` to view an AWS Elemental MediaTailor configuration, the configuration must be tagged `Owner=richard-roe` or `owner=richard-roe`. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

# Troubleshooting AWS Elemental MediaTailor Identity and Access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS Elemental MediaTailor and IAM.

## I Am Not Authorized to Perform an Action in AWS Elemental MediaTailor

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a `configuration` but does not have `mediatailor:ListPlaybackConfigurations` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 mediatailor:ListPlaybackConfigurations on resource: my-example-configuration
```

In this case, Mateo asks his administrator to update his policies to allow him to access configuration resources.

## I Am Not Authorized to Perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to AWS Elemental MediaTailor.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS Elemental MediaTailor. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

## I Want to View My Access Keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing Access Keys in the *IAM User Guide*.

## I'm an Administrator and Want to Allow Others to Access AWS Elemental MediaTailor

To allow others to access AWS Elemental MediaTailor, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in MediaTailor.

To get started right away, see Creating Your First IAM Delegated User and Group in the *IAM User Guide*.

See also Creating a Non-Admin IAM User (p. 6).

## I Want to Allow People Outside of My AWS Account to Access My AWS Elemental MediaTailor Resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support

resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS Elemental MediaTailor supports these features, see How AWS Elemental MediaTailor Works with IAM (p. 88).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing Access to an IAM User in Another AWS Account That You Own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing Access to AWS Accounts Owned by Third Parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing Access to Externally Authenticated Users (Identity Federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the *IAM User Guide*.

# Logging and Monitoring in AWS Elemental MediaTailor

This section provides an overview of the options for logging and monitoring in AWS Elemental MediaTailor for security purposes. For more information about logging and monitoring in MediaTailor see Monitoring and Troubleshooting AWS Elemental MediaTailor (p. 97).

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Elemental MediaTailor and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. AWS provides several tools for monitoring your MediaTailor resources and responding to potential incidents:

## Amazon CloudWatch Alarms

Using CloudWatch alarms, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon SNS topic or AWS Auto Scaling policy. CloudWatch alarms don't invoke actions because they are in a particular state. Rather, the state must have changed and been maintained for a specified number of periods. For more information, see the section called “Monitoring with CloudWatch Metrics” (p. 98).

## AWS CloudTrail Logs

CloudTrail provides a record of actions taken by a user, role, or an AWS service in AWS Elemental MediaTailor. Using the information collected by CloudTrail, you can determine the request that was made to MediaTailor, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging AWS Elemental MediaTailor API Calls with AWS CloudTrail (p. 126).

## AWS Trusted Advisor

Trusted Advisor draws upon best practices learned from serving hundreds of thousands of AWS customers. Trusted Advisor inspects your AWS environment and then makes recommendations when opportunities exist to save money, improve system availability and performance, or help close security gaps. All AWS customers have access to five Trusted Advisor checks. Customers with a Business or Enterprise support plan can view all Trusted Advisor checks.

For more information, see AWS Trusted Advisor.

# Compliance Validation for AWS Elemental MediaTailor

Third-party auditors assess the security and compliance of AWS Elemental MediaTailor as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using MediaTailor is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. If your use of MediaTailor is subject to compliance with standards such as HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Config – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

# Resilience in AWS Elemental MediaTailor

The AWS global infrastructure is built around Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

# Infrastructure Security in AWS Elemental MediaTailor

As a managed service, AWS Elemental MediaTailor is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access MediaTailor through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2. Clients must also support cipher

suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

# Monitoring and Troubleshooting AWS Elemental MediaTailor

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Elemental MediaTailor and your other AWS solutions. AWS provides the following monitoring tools to watch MediaTailor, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and the applications that you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the Amazon CloudWatch User Guide.
- *Amazon CloudWatch Logs* enables you to monitor, store, and access your log files from all interactions with your ad decision server (ADS). AWS Elemental MediaTailor emits logs for ad requests, redirects, responses, and reporting requests and responses. Errors from the ADS and origin servers are also emitted to log groups in Amazon CloudWatch. You can also archive your log data in highly durable storage. For general information, see the Amazon CloudWatch Logs User Guide. For information on the ADS logs and how to access them for analysis through Amazon CloudWatch Logs Insights, see Viewing and Querying AWS Elemental MediaTailor ADS Logs (p. 101).

**Topics**



## Setting Up Permissions for Amazon CloudWatch

Use AWS Identity and Access Management (IAM) to create a role that gives AWS Elemental MediaTailor access to Amazon CloudWatch. You must perform these steps for CloudWatch Logs to be published for your account. CloudWatch automatically publishes metrics for your account.

**To allow MediaTailor access to CloudWatch**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
3. Choose the **Another AWS account** role type.
4. For **Account ID**, enter your AWS account ID.
5. Select **Require external ID** and enter `Midas`. This option automatically adds a condition to the trust policy that allows the service to assume the role only if the request includes the correct `sts:ExternalID`.
6. Choose **Next: Permissions**.
7. Add a permissions policy that specifies what actions this role can complete. Select from one of the following options, and then choose **Next: Review**:

- **CloudWatchLogsFullAccess** to provide full access to Amazon CloudWatch Logs
- **CloudWatchFullAccess** to provide full access to Amazon CloudWatch

8. For **Role name**, enter **`MediaTailorLogger`**, and then choose **Create role**.
9. On the **Roles** page, choose the role that you just created.
10. To update the principal, edit the trust relationship:

    1. On the role's **Summary** page, choose the **Trust relationship** tab.
    2. Choose **Edit trust relationship**.
    3. In the policy document, change the principal to the MediaTailor service. It should look like this:

    ```
    "Principal": {
       "Service": "mediatailor.amazonaws.com"
    },
    ```

    The entire policy should read as follows:

    ```
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "mediatailor.amazonaws.com"
          },
          "Action": "sts:AssumeRole",
          "Condition": {
            "StringEquals": {
              "sts:ExternalId": "Midas"
            }
          }
        }
      ]
    }
    ```

    4. Choose **Update Trust Policy**.

# Monitoring AWS Elemental MediaTailor with Amazon CloudWatch Metrics

You can monitor AWS Elemental MediaTailor metrics using CloudWatch. CloudWatch collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the Amazon CloudWatch User Guide.

Metrics are grouped first by the service namespace, and then by the various dimension combinations within each namespace.

### To view metrics using the CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Under **All metrics**, choose the **MediaTailor** namespace.

4. Select the metric dimension to view the metrics (for example, **originID**).
5. Specify the time period that you want to view.

### To view metrics using the AWS CLI

- At a command prompt, use the following command:

```
aws cloudwatch list-metrics --namespace "AWS/MediaTailor"
```

# AWS Elemental MediaTailor CloudWatch Metrics

The AWS Elemental MediaTailor namespace includes the following metrics. These metrics are published by default to your account.

| Metric | Description |
|---|---|
| `AdDecisionServer.Ads` | The count of ads included in ad decision server (ADS) responses for the time period that you specified. |
| `AdDecisionServer.Duration` | The total duration, in milliseconds, of all ads that MediaTailor received from the ADS for the time period that you specified. |
| `AdDecisionServer.Errors` | The number of non-HTTP 200 status code responses, empty responses, and timed-out responses that MediaTailor received from the ADS in the time period that you specified. |
| `AdDecisionServer.FillRate` | The simple average of the rates at which the responses from the ADS filled the corresponding individual ad avails for the time period that you specified.<br><br>To get the weighted average, calculate the `AdDecisionServer.Duration` as a percentage of the `Avail.Duration`. For more information about simple and weighted averages, see Simple and Weighted Averages (p. 100). |
| `AdDecisionServer.Timeouts` | The number of timed-out requests to the ADS in the time period that you specified. |
| `AdNotReady` | The number of times that the ADS pointed at an ad that wasn't yet transcoded by the internal transcoder service in the time period that you specified.<br><br>A high value for this metric might contribute to a low overall `Avail.FillRate`. |
| `Avail.Duration` | The total duration, in milliseconds, of all ad avails that MediaTailor encountered in the time period that you specified. |

| Metric | Description |
|---|---|
| `Avail.FilledDuration` | The total duration, in milliseconds, of ad avail time that MediaTailor filled with ads in the time period that you specified. |
| `Avail.FillRate` | The simple average of the rates at which MediaTailor filled the individual ad avails for the time period that you specified.<br><br>To get the weighted average, calculate the `Avail.FilledDuration` as a percentage of the `Avail.Duration`. For more information about simple and weighted averages, see Simple and Weighted Averages (p. 100).<br><br>The maximum `Avail.FillRate` that MediaTailor can attain is bounded by the `AdDecisionServer.FillRate`. If the `Avail.FillRate` is low, compare it to the `AdDecisionServer.FillRate`. If the `AdDecisionServer.FillRate` is low, your ADS might not be returning enough ads for the avail durations. |
| `GetManifest.Errors` | The number of errors received while MediaTailor was generating manifests in the time period that you specified. |
| `Origin.Errors` | The number of non-HTTP 200 status code responses and timed-out responses that MediaTailor received from the origin server in the time period that you specified. |
| `Origin.Timeouts` | The number of timed-out requests to the origin server in the time period that you specified. |

## Simple and Weighted Averages

You can retrieve the simple average and the weighted average for the responses from the ADS to ad requests from MediaTailor and for how MediaTailor fills ad avails:

- The *simple averages* are provided in the `AdDecisionServer.FillRate` and the `Avail.FillRate`. These are the averages of the fill rate percentages for the individual avails for the time period. The simple averages don't take into account any differences between the durations of the individual avails.
- The *weighted averages* are the fill rate percentages for the sum of all avail durations. These are calculated as (`AdDecisionServer.Duration`*100)/`Avail.Duration` and (`Avail.FilledDuration`*100)/`Avail.Duration`. These averages reflect the differences in duration of each ad avail, giving more weight to those with longer duration.

For a time period that contains just a single ad avail, the simple average provided by the `AdDecisionServer.FillRate` is equal to the weighted average provided by (`AdDecisionServer.Duration`*100)/`Avail.Duration`. The simple average provided by the `Avail.FillRate` is equal to the weighted average provided by (`Avail.FilledDuration`*100)/`Avail.Duration`.

**Example**

Assume the time period that you specified has the following two ad avails:

- The first ad avail has 90 seconds duration:
  - The ADS response for the avail provides 45 seconds of ads (50% filled).
  - MediaTailor fills 45 seconds worth of the ad time available (50% filled).
- The second ad avail has 120 seconds duration:
  - The ADS response for the avail provides 120 seconds of ads (100% filled).
  - MediaTailor fills 90 seconds worth of the ad time available (75% filled).

The metrics are as follows:

- `Avail.Duration` is 210, the sum of the two ad avail durations: 90 + 120.
- `AdDecisionServer.Duration` is 165, the sum of the two response durations: 45 + 120.
- `Avail.FilledDuration` is 135, the sum of the two filled durations: 45 + 90.
- `AdDecisionServer.FillRate` is 75%, the average of the percentages filled for each avail: (50% + 100%) / 2. This is the simple average.
- The weighted average for the ADS fill rates is 78.57%, which is `AdDecisionServer.Duration` as a percentage of the `Avail.Duration`: (165*100) / 210. This calculation accounts for the differences in the durations.
- `Avail.FillRate` is 62.5%, the average of the filled percentages for each avail: (50% + 75%) / 2. This is the simple average.
- The weighted average for the MediaTailor avail fill rates is 64.29%, which is the `Avail.FilledDuration` as a percentage of the `Avail.Duration`: (135*100) / 210. This calculation accounts for the differences in the durations.

The highest `Avail.FillRate` that MediaTailor can attain for any ad avail is 100%. The ADS might return more ad time than is available in the avail, but MediaTailor can only fill the time available.

## AWS Elemental MediaTailor CloudWatch Dimensions

You can filter the AWS Elemental MediaTailor data using the following dimension.

| Dimension | Description |
| --- | --- |
| `Configuration Name` | Indicates the configuration that the metric belongs to. |

# Viewing and Querying AWS Elemental MediaTailor ADS Logs

You can view and query AWS Elemental MediaTailor ADS logs using Amazon CloudWatch Logs Insights. MediaTailor sends event logs to CloudWatch for normal processing and error conditions. The logs adhere to a JSON schema. Through CloudWatch Logs Insights, you can select logs by time frame, and then run queries against them.

For general information, see Analyze Log Data with CloudWatch Logs Insights.

**Note**
To access the logs, you need permissions to access Amazon CloudWatch. For instructions, see Setting Up Permissions for Amazon CloudWatch (p. 97).

**To view and query ADS logs using the CloudWatch console**

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, under **Logs**, choose **Insights**.
3. In the search bar, enter **`ads_inter`**, and then from the dropdown list select `MediaTailor/AdDecisionServerInteractions`.
4. (Optional) Adjust the time period that you want to study.
5. (Optional) Change the query in the dialog box. For general guidance, see CloudWatch Logs Insights Query Syntax. For examples of queries for MediaTailor ADS, see Querying the ADS Logs (p. 111).
6. Choose **Run query**. The query might take a few seconds, during which time **Cancel** appears in place of **Run query**.
7. (Optional) To export the results as a CSV file, choose **Actions**, and then choose **Download query results (CSV)**.

**Note**
The console limits the number of records that it returns in query results and that it exports, so for bulk data, use the API, AWS CLI, or an SDK.

**Topics**



# ADS Log Description

This section describes the structure and contents of the ADS log description. To explore on your own in a JSON editor, use the listing at the section called "ADS Log JSON Schema" (p. 112).

Each event in the ADS log contains the standard fields that are generated by CloudWatch Logs. For information, see Analyze Log Data with CloudWatch Logs Insights.

## ADS Logs Properties

This section describes the properties of the ADS logs.

**ADS Logs Properties**

| Property | Type | Required | Description |
|---|---|---|---|
| `adsRequestUrl` | string | false | The full URL of the ADS request made by MediaTailor. |
| `avail` | object of type avail (p. 105) | false | Information about an avail that MediaTailor fills with ads. Currently, for the `FILLED_AVAIL` event type, this is the plan created by MediaTailor when it |

| Property | Type | Required | Description |
|---|---|---|---|
| | | | first encounters the avail. How the avail is eventually filled may vary from this plan, depending on how the content plays out. |
| `awsAccountId` | string | true | The AWS account ID for the MediaTailor configuration that was used for the session. |
| `customerId` | string | true | The hashed version of the AWS account ID, which you can use to correlate multiple log entries. |
| `eventDescription` | string | true | A short description of the event that triggered this log message, provided by the MediaTailor service. By default, this is empty. Example: `Got VAST response`. |
| `eventTimestamp` | string | true | The date and time of the event. |
| `eventType` | string | true | The code for the event that triggered this log message. Example: `VAST_RESPONSE`. |
| `originId` | string | true | The configuration name from the MediaTailor configuration. This is different from the video content source, which is also part of the configuration. |
| `requestHeaders` | array of type requestheaders (p. 106) | false | The headers that MediaTailor included with the ADS request. Typically, the logs include these when a request to the ADS fails, to help with troubleshooting. |
| `requestId` | string | true | The MediaTailor request ID, which you can use to correlate multiple log entries for the same request. |

| Property | Type | Required | Description |
|---|---|---|---|
| `sessionId` | string | true | The unique numeric identifier that MediaTailor assigned to the player session. All requests that a player makes for a session have the same session ID. Example: `e039fd39-09f0-46b2-aca9-9871cc116cde`. |
| `sessionType` | string (legal values: [DASH, HLS]) | true | The player's stream type. |
| `vastAd` | object of type vastAd (p. 108) | false | Information about a single ad parsed from the VAST response. |
| `vastResponse` | object of type vastResponse (p. 110) | false | Information about the VAST response that MediaTailor received from the ADS. |
| `vodCreativeOffsets` | object of type vodCreativeOffsets (p. 110) | false | A map that indicates the time offsets in the manifest where MediaTailor will insert avails, based on the VMAP response. |
| `vodVastResponseTimeOffset` | number | false | The VMAP specific time offset for VOD ad insertion. |

## adContent

This section describes the properties of the ADS logs adContent.

### ADS Logs adContent Properties

| Property | Type | Required | Description |
|---|---|---|---|
| `adPlaylistUris` | object of type adPlaylistUris (p. 105) | false | The mapping from the origin manifest for a variant to the ad manifest for the variant. For DASH, this contains a single entry, because all variants are represented in a single DASH manifest. |

## adPlaylistUris

This section describes the properties of the ADS logs adPlaylistUris.

### ADS Logs adPlaylistUris Properties

| Property | Type | Required | Description |
|---|---|---|---|
| `<any string>` | string | false | The URL of the ad manifest for the specific variant. |

## avail

This section describes the properties of the ADS logs avail.

### ADS Logs avail Properties

| Property | Type | Required | Description |
|---|---|---|---|
| `availId` | string | true | The unique identifier for this avail. For HLS, this is the media sequence number where the avail begins. For DASH, this is the period ID. |
| `creativeAds` | array of type creativeAd (p. 106) | true | The ads that MediaTailor inserted into the avail. |
| `fillRate` | number | true | The rate at which the ads fill the avail duration, from 0.0 (for 0%) to 1.0 (for 100%). |
| `filledDuration` | number | true | The sum of the durations of all the ads inserted into the avail. |
| `numAds` | number | true | The number of ads that MediaTailor inserted into the avail. |
| `originAvailDuration` | number | true | The duration of the avail as specified in the content stream from the origin (`CUE_OUT` or `SCTE`). |
| `skippedAds` | array of type skippedAd (p. 107) | false | The ads that MediaTailor didn't insert, for reasons like `TRANSCODE_IN_PROGRESS` and `TRANSCODE_ERROR`. |

| Property | Type | Required | Description |
|---|---|---|---|
| slateAd | object of type slateAd (p. 107) | true | Information about the slate ad, which MediaTailor uses to fill any unfilled segments in the avail. |

## creativeAd

This section describes the properties of the ADS logs creativeAd.

### ADS Logs creativeAd Properties

| Property | Type | Required | Description |
|---|---|---|---|
| adContent | object of type adContent (p. 104) | true | Information about the content of the inserted ad. |
| creativeUniqueId | string | true | The unique identifier for the ad, used as a key for transcoding. This is the ID field for the creative in the VAST response, if available. Otherwise, it's the mezzanine URL of the ad. |
| trackingEvents | object of type trackingEvents (p. 108) | true | The tracking beacon URLs for the various tracking events for the ad. The keys are the event names, and the values are a list of beacon URLs. |
| transcodedAdDuration | number | true | The duration of the ad, calculated from the transcoded asset. |
| uri | string | true | The URL of the mezzanine version of the ad, which is the input to the transcoder. |
| vastDuration | number | true | The duration of the ad, as parsed from the VAST response. |

## requestheaders

This section describes the properties of the ADS logs requestheaders.

### ADS Logs requestheaders Properties

| Property | Type | Required | Description |
|---|---|---|---|
| `name` | string | true | The name of the header. |
| `value` | string | true | The value of the header. |

## skippedAd

This section describes the properties of the ADS logs skippedAd.

### ADS Logs skippedAd Properties

| Property | Type | Required | Description |
|---|---|---|---|
| `adMezzanineUrl` | string | true | The mezzanine URL of the skipped ad. |
| `creativeUniqueId` | string | true | The unique identifier for the ad, used as a key for transcoding. This is the ID field for the creative in the VAST response, if available. Otherwise, it's the mezzanine URL of the ad. |
| `skippedReason` | string | true | The code that indicates why the ad wasn't inserted. Example: `TRANSCODE_IN_PROGRESS`. |
| `transcodedAdDuration` | number | false | The duration of the ad, calculated from the transcoded asset. |
| `vastDuration` | number | true | The duration of the ad, as parsed from the VAST response. |

## slateAd

This section describes the properties of the ADS logs slateAd.

### ADS Logs slateAd Properties

| Property | Type | Required | Description |
|---|---|---|---|
| `adContent` | object of type adContent (p. 104) | true | Information about the content of the inserted ad. |
| `creativeUniqueId` | string | true | The unique identifier for the ad, used as a key |

| Property | Type | Required | Description |
|---|---|---|---|
| | | | for transcoding. This is the ID field for the creative in the VAST response, if available. Otherwise, it's the mezzanine URL of the ad. |
| `transcodedAdDuration` | number | true | The duration of the ad, calculated from the transcoded asset. |
| `uri` | string | true | The URL of the mezzanine version of the ad, which is the input to the transcoder. |

## trackingEvents

This section describes the properties of the ADS logs trackingEvents.

**ADS Logs trackingEvents Properties**

| Property | Type | Required | Description |
|---|---|---|---|
| `<any string>` | array of type string | false | The list of beacon URLs for the specified tracking event (impression, complete, and so on) |

## vastAd

This section describes the properties of the ADS logs vastAd.

**ADS Logs vastAd Properties**

| Property | Type | Required | Description |
|---|---|---|---|
| `adSystem` | string | true | The value of the `AdSystem` tag in the VAST response. |
| `adTitle` | string | true | The media files that are available for the ad in the VAST response. |
| `creativeAdId` | string | true | The value of the adId attribute of the `Creative` tag in the VAST response. |
| `creativeId` | string | true | The value of the id attribute of the |

| Property | Type | Required | Description |
|---|---|---|---|
| | | | `Creative` tag in the VAST response. |
| `duration` | number | true | The approximate duration of the ad, based on the `duration` tag in the `linear` element of the VAST response. |
| `trackingEvents` | object of type trackingEvents (p. 108) | true | The tracking beacon URLs for the various tracking events for the ad. The keys are the event names, and the values are a list of beacon URLs. |
| `vastAdId` | string | true | The value of the id attribute of the `Ad` tag in the VAST response |
| `vastAdTagUri` | string | false | The VMAP-specific redirect URI for an ad. |
| `vastMediaFiles` | array of type vastMediaFile (p. 109) | true | The list of available media files for the ad in the VAST response. |

## vastMediaFile

This section describes the properties of the ADS logs vastMediaFile.

**ADS Logs vastMediaFile Properties**

| Property | Type | Required | Description |
|---|---|---|---|
| `apiFramework` | string | true | The API framework needed to manage the media file. Example: `VPAID`. |
| `bitrate` | number | true | The bitrate of the media file. |
| `delivery` | string | true | The protocol used for the media file, set to either progressive or streaming. |
| `height` | number | true | The pixel height of the media file. |
| `id` | string | true | The value of the id attribute of the `MediaFile` tag. |

| Property | Type | Required | Description |
|---|---|---|---|
| `type` | string | true | The MIME type of the media file, taken from the type attribute of the `MediaFile` tag. |
| `uri` | string | true | The URL of the mezzanine version of the ad, which is the input to the transcoder. |
| `width` | number | true | The pixel width of the media file. |

## vastResponse

This section describes the properties of the ADS logs vastResponse.

**ADS Logs vastResponse Properties**

| Property | Type | Required | Description |
|---|---|---|---|
| `errors` | array of type string | true | The error URLs parsed from the `Error` tags in the VAST response. |
| `vastAds` | array of type vastAd (p. 108) | true | The ads parsed from the VAST response. |
| `version` | string | true | The VAST specification version, parsed from the `version` attribute of the `VAST` tag in the response. |

## vodCreativeOffsets

This section describes the properties of the ADS logs vodCreativeOffsets.

**ADS Logs vodCreativeOffsets Properties**

| Property | Type | Required | Description |
|---|---|---|---|
| `<any string>` | array of type vodCreativeOffset (p. 110) | false | A mapping from a time offset in the manifest to a list of ads to insert at this time. |

## vodCreativeOffset

This section describes the properties of the ADS logs vodCreativeOffset.

**ADS Logs vodCreativeOffset Properties**

| Property | Type | Required | Description |
|---|---|---|---|
| `adContent` | object of type adContent (p. 104) | true | Information about the content of the inserted ad. |
| `creativeUniqueId` | string | true | The unique identifier for the ad, used as a key for transcoding. This is the ID field for the creative in the VAST response, if available. Otherwise, it's the mezzanine URL of the ad. |
| `trackingEvents` | object of type trackingEvents (p. 108) | true | The tracking beacon URLs for the various tracking events for the ad. The keys are the event names, and the values are a list of beacon URLs. |
| `transcodedAdDuration` | number | true | The duration of the ad, calculated from the transcoded asset. |
| `uri` | string | true | The URL of the mezzanine version of the ad, which is the input to the transcoder. |
| `vastDuration` | number | true | The duration of the ad, as parsed from the VAST response. |

# Querying the ADS Logs

CloudWatch Logs Insights provides a rich set of options for querying your logs. For detailed information about querying syntax, see CloudWatch Logs Insights Query Syntax. This section provides examples of common queries to get you started with your ADS logs queries. All queries run against the logs for the current time range setting.

The following query retrieves all information from the ADS logs.

```
fields @timestamp, eventType, sessionId, requestId, @message
| sort sessionId, @timestamp asc
```

The following query retrieves all requests to the ADS. This query shows a way to retrieve the request header contents for MediaTailor logs.

```
fields @timestamp, adsRequestUrl, requestHeaders.0.value as @userAgent,
 requestHeaders.1.value as @xForwardedFor, sessionId, requestId
```

```
| filter eventType = "MAKING_ADS_REQUEST"
| sort @timestamp asc
```

The following query retrieves the ads MediaTailor inserted for a given session.

```
fields @timestamp, sessionId, requestId, @message
| filter eventType = "FILLED_AVAIL"
| sort @timestamp asc
```

The following query retrieves the tracking URLs that MediaTailor called on behalf of the player.

```
fields @timestamp, beaconInfo.trackingEvent, beaconInfo.beaconUri,
 beaconInfo.headers.0.value as @userAgent, beaconInfo.headers.1.value as @xForwardedFor,
 sessionId, requestId
| filter eventType = "BEACON_FIRED"
| sort @timestamp asc
```

The following query retrieves information for a specific playback session, by filtering the results by `sessionId`.

```
fields @timestamp, eventType, sessionId, requestId, @message
| filter sessionId = "0aaf6507-c6f9-4884-bfe7-f2f841cb8195"
| sort @timestamp asc
```

The following query retrieves information for a single request, by filtering the results by `requestId`.

```
fields @timestamp, eventType, sessionId, requestId, @message
| filter requestId = "f5d3cf39-6258-4cf1-b3f6-a34ff8bf641d"
| sort @timestamp asc
```

The following query retrieves a count of log entries for each event type that was logged.

```
fields eventType
| stats count() as @eventCount by eventType
```

The following query retrieves the avail ID and list of skipped ads for all avails that had skipped ads.

```
fields avail.availId
| parse @message '"skippedAds":[*]' as @skippedAdsList
| filter ispresent(@skippedAdsList)
```

# ADS Log JSON Schema

The following lists the JSON schema for the AWS Elemental MediaTailor ADS log.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://amazon.com/elemental/midas/mms/adsLogSchema.json",
  "type": "object",
  "title": "AWS Elemental MediaTailor ADS Log JSON Schema",
  "required": [
    "eventType",
    "eventTimestamp",
    "requestId",
```

```
    "sessionType",
    "eventDescription",
    "awsAccountId",
    "customerId",
    "originId",
    "sessionId"
  ],
  "additionalProperties": false,
  "properties": {
    "eventType": {
      "$id": "#/properties/eventType",
      "type": "string",
      "description": "The code for the event that triggered this log message. Example:
<code>VAST_RESPONSE</code>.",
      "examples": [
        "FILLED_AVAIL"
      ]
    },
    "eventTimestamp": {
      "$id": "#/properties/eventTimestamp",
      "type": "string",
      "description": "The date and time of the event.",
      "examples": [
        "1970-01-01T00:00:00Z"
      ],
      "format": "date-time"
    },
    "requestId": {
      "$id": "#/properties/requestId",
      "type": "string",
      "description": "The MediaTailor request ID, which you can use to correlate multiple
log entries for the same request.",
      "examples": [
        "c7c7ae8c-a61e-44e0-8efd-7723995337a1"
      ],
      "pattern": "^(.*)$"
    },
    "sessionType": {
      "$id": "#/properties/sessionType",
      "type": "string",
      "enum": [
        "HLS",
        "DASH"
      ],
      "description": "The player's stream type."
    },
    "eventDescription": {
      "$id": "#/properties/eventDescription",
      "type": "string",
      "description": "A short description of the event that triggered this log message,
provided by the MediaTailor service. By default, this is empty. Example: <code>Got VAST
response</code>.",
      "default": "",
      "examples": [
        "Got VAST response"
      ],
      "pattern": "^(.*)$"
    },
    "awsAccountId": {
      "$id": "#/properties/awsAccountId",
      "type": "string",
      "description": "The AWS account ID for the MediaTailor configuration that was used
for the session."
    },
    "customerId": {
      "$id": "#/properties/customerId",
```

```
      "type": "string",
      "description": "The hashed version of the AWS account ID, which you can use to
correlate multiple log entries.",
      "pattern": "^(.*)$"
    },
    "originId": {
      "$id": "#/properties/originId",
      "type": "string",
      "description": "The configuration name from the MediaTailor configuration. This is
different from the video content source, which is also part of the configuration.",
      "examples": [
        "external-canary-dash-serverside-reporting-onebox"
      ],
      "pattern": "^(.*)$"
    },
    "sessionId": {
      "$id": "#/properties/sessionId",
      "type": "string",
      "description": "The unique numeric identifier that MediaTailor assigned to the player
session. All requests that a player makes for a session have the same session ID. Example:
<code>e039fd39-09f0-46b2-aca9-9871cc116cde</code>.",
      "examples": [
        "120b9873-c007-40c8-b3db-0f1bd194970b"
      ],
      "pattern": "^(.*)$"
    },
    "avail": {
      "$id": "#/properties/avail",
      "type": "object",
      "title": "avail",
      "description": "Information about an avail that MediaTailor fills with ads.
Currently, for the <code>FILLED_AVAIL</code> event type, this is the plan created by
MediaTailor when it first encounters the avail. How the avail is eventually filled may
vary from this plan, depending on how the content plays out.  ",
      "required": [
        "creativeAds",
        "originAvailDuration",
        "filledDuration",
        "fillRate",
        "numAds",
        "slateAd",
        "availId"
      ],
      "additionalProperties":  false,
      "properties": {
        "originAvailDuration": {
          "$id": "#/properties/avail/originAvailDuration",
          "type": "number",
          "description": "The duration of the avail as specified in the content stream from
the origin (<code>CUE_OUT</code> or <code>SCTE</code>)."
        },
        "filledDuration": {
          "$id": "#/properties/avail/filledDuration",
          "type": "number",
          "description": "The sum of the durations of all the ads inserted into the avail."
        },
        "fillRate": {
          "$id": "#/properties/avail/fillRate",
          "type": "number",
          "description": "The rate at which the ads fill the avail duration, from 0.0 (for
0%) to 1.0 (for 100%)."
        },
        "creativeAds": {
          "$id": "#/properties/avail/creativeAds",
          "type": "array",
          "description": "The ads that MediaTailor inserted into the avail.",
```

```
          "items": {
            "type": "object",
            "title": "creativeAd",
            "description": "Information about a single inserted ad.",
            "required": [
              "uri",
              "creativeUniqueId",
              "adContent",
              "trackingEvents",
              "vastDuration",
              "transcodedAdDuration"
            ],
            "additionalProperties": false,
            "properties": {
              "uri": { "$ref": "#/definitions/adMezzanineUri" },
              "creativeUniqueId": { "$ref": "#/definitions/creativeUniqueId" },
              "adContent": { "$ref": "#/definitions/adContent" },
              "trackingEvents": { "$ref": "#/definitions/trackingEvents" },
              "vastDuration": { "$ref": "#/definitions/vastDuration" },
              "transcodedAdDuration": { "$ref": "#/definitions/transcodedAdDuration" }
            }
          }
        },
        "numAds": {
          "$id": "#/properties/avail/numAds",
          "type": "number",
          "description": "The number of ads that MediaTailor inserted into the avail."
        },
        "slateAd": {
          "$id": "#/properties/avail/slateAd",
          "type": ["object", "null"],
          "title": "slateAd",
          "description": "Information about the slate ad, which MediaTailor uses to fill
any unfilled segments in the avail.",
          "additionalProperties": false,
          "required": [
            "uri",
            "creativeUniqueId",
            "adContent",
            "transcodedAdDuration"
          ],
          "properties": {
            "uri": { "$ref": "#/definitions/adMezzanineUri" },
            "creativeUniqueId": { "$ref": "#/definitions/creativeUniqueId" },
            "adContent": { "$ref": "#/definitions/adContent" },
            "transcodedAdDuration": { "$ref": "#/definitions/transcodedAdDuration" }
          }
        },
        "availId": {
          "$id": "#/properties/avail/availId",
          "type": "string",
          "description": "The unique identifier for this avail. For HLS, this is the media
sequence number where the avail begins. For DASH, this is the period ID."
        },
        "skippedAds": {
          "$id": "#/properties/avail/skippedAds",
          "type": "array",
          "description": "The ads that MediaTailor didn't insert, for reasons like
<code>TRANSCODE_IN_PROGRESS</code> and <code>TRANSCODE_ERROR</code>.",
          "items": {
            "type": "object",
            "title": "skippedAd",
            "description": "Information about a single skipped ad.",
            "required": [
              "creativeUniqueId",
              "adMezzanineUrl",
```

```
              "skippedReason",
              "vastDuration"
            ],
            "additionalProperties": false,
            "properties": {
              "creativeUniqueId": { "$ref": "#/definitions/creativeUniqueId" },
              "adMezzanineUrl": {
                "type": "string",
                "description": "The mezzanine URL of the skipped ad."
              },
              "skippedReason": {
                "type": "string",
                "description": "The code that indicates why the ad wasn't inserted.
 Example: <code>TRANSCODE_IN_PROGRESS</code>."
              },
              "vastDuration": { "$ref": "#/definitions/vastDuration" },
              "transcodedAdDuration": { "$ref": "#/definitions/transcodedAdDuration" }
            }
          }
        }
      }
    },

    "vastResponse": {
      "$id": "#/properties/vastResponse",
      "type": "object",
      "title": "vastResponse",
      "description": "Information about the VAST response that MediaTailor received from
 the ADS.",
      "required": [
        "version",
        "vastAds",
        "errors"
      ],
      "additionalProperties":  false,
      "properties": {
        "version": {
          "$id": "#/properties/vastResponse/version",
          "type": "string",
          "description": "The VAST specification version, parsed from the <code>version</
code> attribute of the <code>VAST</code> tag in the response.",
          "examples": [
            "3.0"
          ],
          "pattern": "^(.*)$"
        },
        "vastAds": {
          "$id": "#/properties/vastResponse/vastAds",
          "type": "array",
          "description": "The ads parsed from the VAST response.",
          "items": {
            "$ref": "#/definitions/vastAd"
          }
        },
        "errors": {
          "$id": "#/properties/vastResponse/errors",
          "type": "array",
          "description": "The error URLs parsed from the <code>Error</code> tags in the
 VAST response.",
          "items": {
            "type": "string",
            "description": "A single error URL."
          }
        }
      }
    },
```

```
   "vastAd": {
     "$ref": "#/definitions/vastAd"
   },

   "vodVastResponseTimeOffset": {
     "$id": "#/properties/vodVastResponseTimeOffset",
     "type": "number",
     "description": "The VMAP specific time offset for VOD ad insertion.",
     "examples": [
       5.0
     ]
   },

   "vodCreativeOffsets": {
     "$id": "#/properties/vodCreativeOffsets",
     "type": "object",
     "title": "vodCreativeOffsets",
     "description": "A map that indicates the time offsets in the manifest where
MediaTailor will insert avails, based on the VMAP response.",
     "additionalProperties": {
       "type": "array",
       "$id": "#/properties/vodCreativeOffsets/entry",
       "description": "A mapping from a time offset in the manifest to a list of ads to
insert at this time.",
       "items": {
         "type": "object",
         "$id": "#/properties/vodCreativeOffsets/entry/items",
         "title": "vodCreativeOffset",
         "description": "The list of ads to insert at the specified time offset.",
         "additionalProperties": false,
         "required": [
           "uri",
           "creativeUniqueId",
           "vastDuration",
           "transcodedAdDuration",
           "adContent",
           "trackingEvents"
         ],
         "properties": {
           "uri": { "$ref": "#/definitions/adMezzanineUri" },
           "creativeUniqueId": { "$ref": "#/definitions/creativeUniqueId" },
           "vastDuration": { "$ref": "#/definitions/vastDuration" },
           "transcodedAdDuration": { "$ref": "#/definitions/transcodedAdDuration" },
           "adContent": { "$ref": "#/definitions/adContent" },
           "trackingEvents": { "$ref": "#/definitions/trackingEvents" }
         }
       }
     }
   },

   "adsRequestUrl": {
     "$id": "#/properties/adsRequestUrl",
     "type": "string",
     "description": "The full URL of the ADS request made by MediaTailor."
   },

   "requestHeaders": {
     "$id": "#/properties/requestHeaders",
     "type": "array",
     "description": "The headers that MediaTailor included with the ADS request.
Typically, the logs include these when a request to the ADS fails, to help with
troubleshooting.",
     "items": {
       "type": "object",
       "title": "requestheaders",
```

```
        "description": "The name and value for a single header included in the ADS
 request.",
        "required": [
          "name",
          "value"
        ],
        "additionalProperties": false,
        "properties": {
          "name": {
            "type": "string",
            "description": "The name of the header."
          },
          "value": {
            "type": "string",
            "description": "The value of the header."
          }
        }
      }
    }
  },

  "__COMMENT_oneOf": "The oneOf section defines subtypes for our events. Subtypes can have
 different rules, including which fields are required. For more information, see https://
json-schema.org/understanding-json-schema/reference/combining.html#oneof ",

  "oneOf": [
    { "$ref": "#/definitions/eventMakingAdsRequest" },
    { "$ref": "#/definitions/eventVastResponse" },
    { "$ref": "#/definitions/eventFilledAvail" },
    { "$ref": "#/definitions/eventErrorFiringBeaconFailed" },
    { "$ref": "#/definitions/eventWarningNoAdvertisements" },
    { "$ref": "#/definitions/eventUnknownHost" },
    { "$ref": "#/definitions/eventErrorAdsTimeout" },
    { "$ref": "#/definitions/eventPlannedAvail" },
    { "$ref": "#/definitions/eventEmptyVastResponse" },
    { "$ref": "#/definitions/eventErrorUnknown" },
    { "$ref": "#/definitions/eventVastRedirect" },
    { "$ref": "#/definitions/eventRedirectedVastResponse" },
    { "$ref": "#/definitions/eventErrorAdsResponseParse" },
    { "$ref": "#/definitions/eventErrorAdsInvalidResponse" },
    { "$ref": "#/definitions/eventErrorDisallowedHost"},
    { "$ref": "#/definitions/eventWarningDynamicVariableSubFailed"},
    { "$ref": "#/definitions/eventVodTimeBasedAvailPlanVastResponseForOffset" },
    { "$ref": "#/definitions/eventVodTimeBasedAvailPlanSuccess" }
  ],


  "definitions": {
    "eventMakingAdsRequest": {
      "$id": "#/definitions/eventMakingAdsRequest",
      "required": [
        "eventType",
        "adsRequestUrl"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "MAKING_ADS_REQUEST"
        }
      }
    },

    "eventVastResponse": {
      "$id": "#/definitions/eventVastResponse",
      "_comment": "NOTE: the vastResponse property should ideally be marked as a required
 field for this event, but unfortunately, in the case of an empty vast response, we
```

```
currently emit an EMPTY_VAST_RESPONSE followed by a VAST_RESPONSE, and the vastResponse
property is not present in the latter. We need to fix this so that we don't emit both of
those events in the empty response case, and update this schema to flag vastResponse as
required for VAST_RESPONSE.",
     "required": [
       "eventType"
     ],
     "properties": {
       "eventType": {
         "type": "string",
         "const": "VAST_RESPONSE"
       }
     }
   },

   "eventFilledAvail": {
     "$id": "#/definitions/eventFilledAvail",
     "required": [
       "eventType",
       "avail"
     ],
     "properties": {
       "eventType": {
         "type": "string",
         "const": "FILLED_AVAIL"
       }
     }
   },

   "eventErrorFiringBeaconFailed": {
     "$id": "#/definitions/eventErrorFiringBeaconFailed",
     "required": [
       "eventType",
       "error",
       "beaconInfo"
     ],
     "properties": {
       "eventType": {
         "type": "string",
         "const": "ERROR_FIRING_BEACON_FAILED"
       }
     }
   },

   "eventWarningNoAdvertisements": {
     "$id": "#/definitions/eventWarningNoAdvertisements",
     "required": [
       "eventType"
     ],
     "_comment": "We should really have a more descriptive error field for these events",
     "properties": {
       "eventType": {
         "type": "string",
         "const": "WARNING_NO_ADVERTISEMENTS"
       }
     }
   },

   "eventUnknownHost": {
     "$id": "#/definitions/eventUnknownHost",
     "required": [
       "eventType",
       "requestHeaders"
     ],
     "properties": {
       "eventType": {
```

```
        "type": "string",
        "const": "ERROR_UNKNOWN_HOST"
      }
    }
  },

  "eventErrorAdsTimeout": {
    "$id": "#/definitions/eventErrorAdsTimeout",
    "required": [
      "eventType",
      "adsRequestUrl",
      "requestHeaders"
    ],
    "properties": {
      "eventType": {
        "type": "string",
        "const": "ERROR_ADS_TIMEOUT"
      }
    }
  },

  "eventPlannedAvail": {
    "$id": "#/definitions/eventPlannedAvail",
    "required": [
      "eventType"
    ],
    "_comment": "TODO: Flesh this out as we implement it",
    "properties": {
      "eventType": {
        "type": "string",
        "const": "PLANNED_AVAIL"
      }
    }
  },

  "eventEmptyVastResponse": {
    "$id": "#/definitions/eventEmptyVastResponse",
    "required": [
      "eventType"
    ],
    "properties": {
      "eventType": {
        "type": "string",
        "const": "EMPTY_VAST_RESPONSE"
      }
    }
  },

  "eventErrorUnknown": {
    "$id": "#/definitions/eventErrorUnknown",
    "required": [
      "eventType"
    ],
    "_comment": "TODO: we should have a field for the exception message or something",
    "properties": {
      "eventType": {
        "type": "string",
        "const": "ERROR_UNKNOWN"
      }
    }
  },

  "eventVastRedirect": {
    "$id": "#/definitions/eventVastRedirect",
    "required": [
      "eventType"
```

```
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "VAST_REDIRECT"
        }
      }
    },

    "eventRedirectedVastResponse": {
      "$id": "#/definitions/eventRedirectedVastResponse",
      "required": [
        "eventType"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "REDIRECTED_VAST_RESPONSE"
        }
      },
      "_comment": "NOTE that the property vastResponse is not required because empty vast
responses do not contain a vastResponse."
    },

    "eventErrorAdsResponseParse": {
      "$id": "#/definitions/eventErrorAdsResponseParse",
      "required": [
        "eventType"
      ],
      "_comment": "We should have a field with an error message here",
      "properties": {
        "eventType": {
          "type": "string",
          "const": "ERROR_ADS_RESPONSE_PARSE"
        }
      }
    },

    "eventErrorAdsInvalidResponse": {
      "$id": "#/definitions/eventErrorAdsInvalidResponse",
      "required": [
        "eventType",
        "additionalInfo"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "ERROR_ADS_INVALID_RESPONSE"
        }
      }
    },

    "eventErrorDisallowedHost": {
      "$id": "#/definitions/eventErrorDisallowedHost",
      "required": [
        "eventType"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "ERROR_DISALLOWED_HOST"
        }
      }
    },

    "eventWarningDynamicVariableSubFailed": {
```

```
      "$id": "#/definitions/eventWarningDynamicVariableSubFailed",
      "required": [
        "eventType",
        "adsRequestUrl"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "WARNING_URL_VARIABLE_SUBSTITUTION_FAILED"
        }
      }
    },

    "eventVodTimeBasedAvailPlanVastResponseForOffset": {
      "$id": "#/definitions/eventVodTimeBasedAvailPlanVastResponseForOffset",
      "required": [
        "eventType",
        "vastResponse"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "VOD_TIME_BASED_AVAIL_PLAN_VAST_RESPONSE_FOR_OFFSET"
        }
      }
    },

    "eventVodTimeBasedAvailPlanSuccess": {
      "$id": "#/definitions/eventVodTimeBasedAvailPlanSuccess",
      "required": [
        "eventType",
        "vodCreativeOffsets"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "VOD_TIME_BASED_AVAIL_PLAN_SUCCESS"
        }
      }
    },

    "creativeUniqueId": {
      "type": "string",
      "description": "The unique identifier for the ad, used as a key for transcoding. This
is the ID field for the creative in the VAST response, if available. Otherwise, it's the
mezzanine URL of the ad. "
    },

    "vastDuration": {
      "type": "number",
      "description": "The duration of the ad, as parsed from the VAST response."
    },

    "transcodedAdDuration": {
      "type": "number",
      "description": "The duration of the ad, calculated from the transcoded asset."
    },

    "adContent": {
      "$id": "#/properties/adContent",
      "type": ["object", "null"],
      "title": "adContent",
      "description": "Information about the content of the inserted ad.",
      "additionalProperties": false,
      "properties": {
        "adPlaylistUris": {
```

```
          "$id": "#/properties/adContent/adPlaylistUris",
          "type": "object",
          "title": "adPlaylistUris",
          "description": "The mapping from the origin manifest for a variant to the ad
manifest for the variant. For DASH, this contains a single entry, because all variants are
represented in a single DASH manifest. ",
          "additionalProperties": {
            "$id": "#/properties/adContent/adPlaylistUris/adPlaylistUri",
            "type": "string",
            "description": "The URL of the ad manifest for the specific variant."
          }
        }
      }
    },

    "adMezzanineUri": {
      "type": "string",
      "description": "The URL of the mezzanine version of the ad, which is the input to the
transcoder."
    },

    "trackingEvents": {
      "type": "object",
      "title": "trackingEvents",
      "description": "The tracking beacon URLs for the various tracking events for the ad.
The keys are the event names, and the values are a list of beacon URLs.",

      "additionalProperties":  {
        "type": "array",
        "description": "The list of beacon URLs for the specified tracking event
(impression, complete, and so on)",
        "items": {
          "type": "string",
          "description": "The beacon URLs for this tracking event."
        }
      }
    },

    "vastAd": {
      "$id": "#/properties/vastAd",
      "type": "object",
      "title": "vastAd",
      "description": "Information about a single ad parsed from the VAST response.",
      "required": [
        "vastAdId",
        "adSystem",
        "adTitle",
        "creativeId",
        "creativeAdId",
        "duration",
        "vastMediaFiles",
        "trackingEvents"
      ],
      "additionalProperties":  false,
      "properties": {
        "vastAdId": {
          "$id": "#/properties/vastAd/vastAdId",
          "type": "string",
          "description": "The value of the id attribute of the <code>Ad</code> tag in the
VAST response",
          "examples": [
            "ad1"
          ]
        },
        "adSystem": {
          "$id": "#/properties/vastAd/adSystem",
```

```
          "type": "string",
          "description": "The value of the <code>AdSystem</code> tag in the VAST
 response.",
          "examples": [
            "GDFP"
          ]
        },
        "adTitle": {
          "$id": "#/properties/vastAd/adTitle",
          "type": "string",
          "description": "The media files that are available for the ad in the VAST
 response.",
          "examples": [
            "External NCA1C1L1 LinearInlineSkippable"
          ]
        },
        "creativeId": {
          "$id": "#/properties/vastAd/creativeId",
          "type": "string",
          "description": "The value of the id attribute of the <code>Creative</code> tag in
 the VAST response.",
          "examples": [
            "creative1"
          ]
        },
        "creativeAdId": {
          "$id": "#/properties/vastAd/creativeAdId",
          "type": "string",
          "description": "The value of the adId attribute of the <code>Creative</code> tag
 in the VAST response."
        },
        "duration": {
          "$id": "#/properties/vastAd/duration",
          "type": "number",
          "description": "The approximate duration of the ad, based on the <code>duration</
code> tag in the <code>linear</code> element of the VAST response.",
          "examples": [
            30,
            30.0
          ]
        },
        "vastMediaFiles": {
          "$id": "#/properties/vastAd/vastMediaFiles",
          "type": "array",
          "description": "The list of available media files for the ad in the VAST
 response.",
          "items": {
            "$id": "#/properties/vastAd/vastMediaFiles/items",
            "type": "object",
            "title": "vastMediaFile",
            "description": "Information about a media file for the ad.",
            "required": [
              "uri",
              "id",
              "delivery",
              "type",
              "apiFramework",
              "width",
              "height",
              "bitrate"
            ],
            "additionalProperties":  false,
            "properties": {
              "uri": { "$ref": "#/definitions/adMezzanineUri" },
              "id": {
                "$id": "#/properties/vastAd/vastMediaFiles/items/properties/id",
```

```
              "type": "string",
              "description": "The value of the id attribute of the <code>MediaFile</code>
tag.",
              "examples": [
                "GDFP"
              ]
            },
            "delivery": {
              "$id": "#/properties/vastAd/vastMediaFiles/items/properties/delivery",
              "type": "string",
              "description": "The protocol used for the media file, set to either
progressive or streaming.",
              "examples": [
                "progressive"
              ]
            },
            "type": {
              "$id": "#/properties/vastAd/vastMediaFiles/items/properties/type",
              "type": "string",
              "description": "The MIME type of the media file, taken from the type
attribute of the <code>MediaFile</code> tag.",
              "examples": [
                "video/mp4"
              ]
            },
            "apiFramework": {
              "$id": "#/properties/vastAd/vastMediaFiles/items/properties/apiFramework",
              "type": "string",
              "description": "The API framework needed to manage the media file. Example:
<code>VPAID</code>."
            },
            "width": {
              "$id": "#/properties/vastAd/vastMediaFiles/items/properties/width",
              "type": "integer",
              "description": "The pixel width of the media file.",
              "examples": [
                1280
              ]
            },
            "height": {
              "$id": "#/properties/vastAd/vastMediaFiles/items/properties/height",
              "type": "integer",
              "description": "The pixel height of the media file.",
              "examples": [
                720
              ]
            },
            "bitrate": {
              "$id": "#/properties/vastAd/vastMediaFiles/items/properties/bitrate",
              "type": "integer",
              "description": "The bitrate of the media file.",
              "examples": [
                533
              ]
            }
          }
        }
      },
      "trackingEvents": { "$ref": "#/definitions/trackingEvents" },
      "vastAdTagUri": {
        "$id": "#/properties/vastAd/vastAdTagUri",
        "type": "string",
        "description": "The VMAP-specific redirect URI for an ad.",
        "examples": [
          "https://ads.redirect.com/redirect1"
        ]
```

```
        }
      }
    }

  }
}
```

# Logging AWS Elemental MediaTailor API Calls with AWS CloudTrail

AWS Elemental MediaTailor is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in MediaTailor. CloudTrail captures all API calls for MediaTailor as events. The calls captured include calls from the MediaTailor console and code calls to the MediaTailor API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for MediaTailor. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to MediaTailor, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

## AWS Elemental MediaTailor Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS Elemental MediaTailor, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for AWS Elemental MediaTailor, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All AWS Elemental MediaTailor actions are logged by CloudTrail and are documented in the *AWS Elemental MediaTailor API Reference*. For example, calls to the `PutPlaybackConfiguration` and `ListPlaybackConfigurations` operations generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials
- Whether the request was made with temporary security credentials for a role or federated user
- Whether the request was made by another AWS service

For more information, see CloudTrail userIdentity Element.

# Understanding AWS Elemental MediaTailor Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `PutPlaybackConfiguration` action:


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAEXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/testuser",
        "accountId": "111122223333",
        "accessKeyId": "AIDAEXAMPLE",
        "userName": "testuser"
    },
    "eventTime": "2018-12-28T22:53:46Z",
    "eventSource": "mediatailor.amazonaws.com",
    "eventName": "PutPlaybackConfiguration",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "1.2.3.4",
    "userAgent": "PostmanRuntime/7.4.0",
    "requestParameters": {
        "VideoContentSourceUrl": "http://examplevideo.com",
        "Name": "examplename",
        "AdDecisionServerUrl": "http://exampleads.com"
    },
    "responseElements": {
        "SessionInitializationEndpointPrefix": "https://
bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com/v1/session/
AKIAIOSFODNN7EXAMPLE/examplename/",
        "DashConfiguration": {
            "ManifestEndpointPrefix": "https://
bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com/v1/dash/
AKIAIOSFODNN7EXAMPLE/examplename/",
            "MpdLocation": "EMT_DEFAULT"
        },
        "AdDecisionServerUrl": "http://exampleads.com",
        "CdnConfiguration": {},
        "PlaybackEndpointPrefix": "https://bdaaeb4bd9114c088964e4063f849065.mediatailor.us-
east-1.amazonaws.com",
        "HlsConfiguration": {
            "ManifestEndpointPrefix": "https://
bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com/v1/master/
AKIAIOSFODNN7EXAMPLE/examplename/"
        },
        "VideoContentSourceUrl": "http://examplevideo.com",
        "Name": "examplename"
    },
    "requestID": "1a2b3c4d-1234-5678-1234-1a2b3c4d5e6f",
    "eventID": "987abc65-1a2b-3c4d-5d6e-987abc654def",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
```

```
}
```

The following example shows a CloudTrail log entry that demonstrates the `GetPlaybackConfiguration` action:

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAEXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/testuser",
        "accountId": "111122223333",
        "accessKeyId": "AIDAEXAMPLE",
        "userName": "testuser"
    },
    "eventTime": "2018-12-28T22:52:37Z",
    "eventSource": "mediatailor.amazonaws.com",
    "eventName": "GetPlaybackConfiguration",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "1.2.3.4",
    "userAgent": "PostmanRuntime/7.4.0",
    "requestParameters": {
        "Name": "examplename"
    },
    "responseElements": null,
    "requestID": "0z1y2x3w-0123-4567-9876-6q7r8s9t0u1v",
    "eventID": "888ddd77-3322-eeww-uuii-abc123jkl343",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

# Limits in AWS Elemental MediaTailor

The following sections provide information about the limits in AWS Elemental MediaTailor. For information about requesting an increase to soft limits, see AWS Service Limits. Hard limits can't be changed.

## Soft Limits

The following table describes limits in AWS Elemental MediaTailor that can be increased. For information about changing limits, see AWS Service Limits.

| Limit | Default Setting | Description |
|---|---|---|
| Transactions | 3,000 | The maximum number of concurrent transactions per second across all request types. This is an account-level limit. Your transaction count depends largely on how often players request updated manifests and the number of players. Each player request counts as a transaction. |

## Hard Limits

The following table describes limits in AWS Elemental MediaTailor that can't be increased.

| Limit | Setting | Description |
|---|---|---|
| Ad decision server (ADS) length | 25,000 | The maximum number of characters in an ad decision server (ADS) specification. |
| Ad decision server (ADS) redirects | 3 | The maximum depth of redirects that MediaTailor follows in VAST wrapper tags. |
| Ad decision server (ADS) timeout | 1.5 | The maximum number of seconds that MediaTailor waits before timing out on an open connection to an ad decision server (ADS). When a connection times out, MediaTailor is unable to fill the ad avail with ads due to no response from the ADS. |
| Configurations | 500 | The maximum number of configurations that MediaTailor allows. |

| Limit | Setting | Description |
| --- | --- | --- |
| Content origin length | 512 | The maximum number of characters in a content origin specification. |
| Content origin server timeout | 2 | The maximum number of seconds that MediaTailor waits before timing out on an open connection to the content origin server when requesting template manifests. Timeouts generate `HTTP 504 (GatewayTimeoutException)` response errors. |
| Manifest size | 2 | The maximum size, in MB, of any playback manifest, whether in input or output. To ensure that you stay under the limit, use `gzip` to compress your input manifests into MediaTailor. |
| Session expiration | 10 times the manifest duration | The maximum amount of time that MediaTailor allows a session to remain inactive before ending the session. Session activity can be a player request or an advance by the origin server. When the session expires, MediaTailor returns an `HTTP 400 (Bad Request)` response error. |

# AWS Elemental MediaTailor Resources

The following table lists related resources that you will find useful as you work with AWS Elemental MediaTailor.

| Resource | Description |
| --- | --- |
| SCTE Standard: SCTE 35 | The SCTE standard document for SCTE35. |
| Classes and Workshops | Links to role-based and specialty courses as well as self-paced labs to help sharpen your AWS skills and gain practical experience. |
| AWS Developer Tools | Links to developer tools, SDKs, IDE tool kits, and command line tools for developing and managing AWS applications. |
| AWS Whitepapers | Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts. |
| AWS Support Center | The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor. |
| AWS Support | The primary webpage for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud. |
| Contact Us | A central contact point for inquiries concerning AWS billing, account, events, abuse, and other issues. |
| AWS Site Terms | Detailed information about our copyright and trademark; your account, license, and site access; and other topics. |

# Document History for AWS Elemental MediaTailor

The following table describes important changes to this documentation.

| update-history-change | update-history-description | update-history-date |
| --- | --- | --- |
| Support for live pre-roll ads (p. 49) | Added support for inserting pre-roll ads at the beginning of a live stream. | September 11, 2019 |
| Analyzing ADS Logs in Amazon CloudWatch Logs Insights (p. 101) | Added information for using the AWS Elemental MediaTailor ADS logs and CloudWatch Logs Insights to analyze your MediaTailor sessions. | August 13, 2019 |
| New security chapter (p. 83) | Added a security chapter to enhance and standardize coverage. | May 23, 2019 |
| DASH single-period manifests (p. 21) | Added support for single-period DASH manifests from the origin server, with multi-period manifest output. | April 4, 2019 |
| Support for SCTE-35 UPIDs in the ADS URL (p. 62) | Added support for including a unique program ID (UPID) in the ad decision server (ADS) URL. This allows the ADS to provide program-level ad targeting within a live linear stream. | March 28, 2019 |
| Client-side reporting supports companion ads (p. 71) | For client-side reporting, the AWS Elemental MediaTailor tracking URL response now includes companion ad metadata. | March 28, 2019 |
| HLS ad marker documentation (p. 18) | Added a section that describes supported HLS ad markers. | March 1, 2019 |
| Tagging support (p. 77) | Added support for tagging of configuration resources in AWS Elemental MediaTailor. Tagging allows you to identify and organize your AWS resources and control access to them and to track your AWS costs. | February 14, 2019 |
| Added AWS CloudTrail logging information (p. 126) | Added topic about using CloudTrail to log actions in the AWS Elemental MediaTailor API. | February 11, 2019 |

| | | |
|---|---|---|
| Added section on playback errors (p. 79) | Added information about the errors that might be returned by MediaTailor during playback, in response to requests from a player or a content delivery network (CDN). | February 4, 2019 |
| DASH base64-encoded binary (p. 21) | Added support for providing splicing information in manifests in base64-encoded binary, inside `<scte35:Signal>` `<scte35:Binary>` markers. | January 4, 2019 |
| DASH time signal (p. 21) | Added support for providing splicing information in manifests inside `<scte35:TimeSignal>` markers. | December 5, 2018 |
| DASH location support (p. 47) | Added support for the MPEG-DASH `<Location>` tag. | December 4, 2018 |
| DASH support (p. 21) | Added support for MPEG-DASH manifests. | November 14, 2018 |
| Updated limits tables (p. 129) | Updated limits for configurations and manifest size. | October 13, 2018 |
| New and updated metrics (p. 98) | Added metrics for ad decision server (ADS) and origin timeouts, and updated the ADS and origin error definitions to include timed-out responses. | October 13, 2018 |
| Better documentation coverage for server-side and client-side ad insertion use cases (p. 60) | Expanded description and examples to cover the use of dynamic ad variables for server-side ad insertion and for client-side ad insertion. | October 1, 2018 |
| New Regions (p. 4) | Added support for the PDX and FRA Regions. | July 18, 2018 |
| VAST/VPAID (p. 57) | Added information about VAST and VPAID. | March 16, 2018 |
| CloudWatch (p. 97) | Added information about available CloudWatch metrics, namespaces, and dimensions. | March 16, 2018 |
| New Regions (p. 4) | Added support for the Asia Pacific (Singapore), Asia Pacific (Sydney), and Asia Pacific (Tokyo) Regions. | February 8, 2018 |
| Default Amazon CloudFront distribution paths (p. 53) | Added the list of paths for the Amazon CloudFront distribution where AWS Elemental MediaTailor stores ads. | February 6, 2018 |

| | | |
|---|---|---|
| IAM policy information (p. 5) | Added IAM policy information specific to AWS Elemental MediaTailor. Added instructions for creating non-admin roles with limited permissions. | January 3, 2018 |
| First release (p. 1) | First release of this documentation. | November 27, 2017 |

**Note**

- The AWS Media Services are not designed or intended for use with applications or in situations requiring fail-safe performance, such as life safety operations, navigation or communication systems, air traffic control, or life support machines in which the unavailability, interruption or failure of the services could lead to death, personal injury, property damage or environmental damage.

# AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.