AWS Elemental MediaTailor

User Guide

AWS Elemental MediaTailor: User Guide

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 other content. The service also offers advanced tracking of ad views, which further increases the monetization of content.

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

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 AWS Elemental 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, AWS Elemental 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 AWS Elemental MediaTailor. The responses dictate what ads MediaTailor inserts in the manifest. VMAP also includes timing for ad breaks. For more information about the logic behind MediaTailor ad insertion, see Ad Behavior in AWS Elemental MediaTailor (p. 47). For more information about how MediaTailor works with VAST, see *VAST* (p. 38).

How AWS Elemental MediaTailor Works

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

Here is the general AWS Elemental 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. 51). For information about configuring the ad targeting parameters, see Dynamic Ad Variables in AWS Elemental MediaTailor (p. 41).
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. 47).
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. 51). 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 AWS Elemental 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 AWS Elemental 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 AWS Elemental MediaTailor

AWS Elemental MediaTailor supports the following features:

Ad Tracking Reporting

AWS Elemental 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. 51).

Audio

AWS Elemental MediaTailor supports multiple audio tracks. For more information, see Alternate Audio and Subtitles (p. 16).

Content and Ad Continuity

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

Personalized Content

AWS Elemental 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 AWS Elemental MediaTailor. For more information about how MediaTailor works with origin servers, see How AWS Elemental 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 AWS Elemental MediaTailor

You can access AWS Elemental 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 AWS Elemental MediaTailor console, use the following link: **`https://console.aws.amazon.com/mediatailor/home`**.

Pricing for AWS Elemental MediaTailor

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

Regions for AWS Elemental MediaTailor

To reduce data latency in your applications, AWS Elemental MediaTailor offers regional endpoints to make your requests. To view the list of Regions in which AWS Elemental 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 AWS Elemental 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. 9)

Sensitive Information

AWS Elemental MediaTailor doesn't require that you supply any customer data.

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.

Transcoded Ad Management

AWS Elemental 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

Before you start using AWS Elemental MediaTailor, complete the following steps.

Topics

Signing Up for AWS

If you do not have an AWS account, use the following procedure to create one.

To sign up for AWS

1. Open https://aws.amazon.com/, and then choose **Create an AWS Account**.

   Note
   If you previously signed in to the AWS Management Console using AWS account root user credentials, choose **Sign in to a different account**. If you previously signed in to the console using IAM credentials, choose **Sign-in using root account credentials**. Then choose **Create a new AWS account**.

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code using 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 IAM user for yourself and add the user to an Administrators group

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 of the console, choose **Users**, and then choose **Add user**.
3. For **User name**, type **`Administrator`**.
4. Select the check box next to **AWS Management Console access**, select **Custom password**, and then type the new user's password in the text box. You can optionally select **Require password reset** to force the user to create a new password the next time the user signs in.
5. Choose **Next: Permissions**.
6. On the **Set permissions** page, choose **Add user to group**.
7. Choose **Create group**.
8. In the **Create group** dialog box, for **Group name** type **`Administrators`**.
9. For **Filter policies**, select the check box for **AWS managed - job function**.
10. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.
11. 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.
12. Choose **Next: Tags** to add metadata to the user by attaching tags as key-value pairs.
13. 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 to restrict users' permissions to specific AWS resources, go to 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: Create Policies

Create two policies for AWS Elemental MediaTailor: one to provide read/write access, and one to provide read-only access. Perform these steps one time only for each policy.

To create policies for AWS Elemental MediaTailor

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Use your Administrator user credentials to sign in to the IAM console.
3. In the navigation pane of the console, choose **Policies**, and then choose **Create policy**.
4. Choose the **JSON** tab and paste the following policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "mediatailor:*",
            "Resource": "*"
        }
    ]
}
```

   This policy allows all actions on all resources in AWS Elemental MediaTailor.
5. Choose **Review policy**.
6. On the **Review policy** page, for **Name**, enter **`MediaTailorAllAccess`**, and then choose **Create policy**.
7. On the **Policies** page, repeat the steps in this section to create a read-only policy. Use the following policy and call it **`MediaTailorReadOnlyAccess`**:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "mediatailor:GetPlaybackConfiguration",
                "mediatailor:ListPlaybackConfigurations"
            ],
            "Resource": "*"
        }
    ]
}
```

Step 2: Create User Groups

Create a user group for each of the policies that you created in step 1. 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**, and then choose **Create New Group**.
2. On the **Set Group Name** page, enter a name for the group, such as **`MediaTailorAdmins`**. Choose **Next Step**.
3. On the **Attach Policy** page, for **Filter**, choose **Customer Managed**.
4. In the policy list, choose the **MediaTailorAllAccess** policy that you created.
5. On the **Review** page, verify that the correct policy is added to this group, and then choose **Create Group**.
6. On the **Groups** page, repeat the steps in this section to create a user group that has read-only permissions. In step 4, choose **MediaTailorReadOnlyAccess**.

Step 3: 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).
- For live content, manifests must contain markers to delineate ad breaks. 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. 47).
- 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.
- Manifests must mark periods as ad breaks 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 break starts at the period start and lasts for the SCTE-35 event duration, if one is specified, or until the next period starts.

  The following example shows a period designated as an ad break using splice insert markers. The duration for this break 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>
```

- Periods marked as ad breaks must have the same `AdaptationSet` and `Representation` settings as content stream periods. 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 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. 41).

Enter the configured "template" URL when you create the origin server/ADS mapping in MediaTailor, in Step 4: Create a Configuration (p. 12).

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. 11), 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. 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:

- HLS: `playback-endpoint/v1/master/hashed-account-id/origin-id/master.m3u8`
- 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. For example:

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

- `hashed-account-id` is your AWS account ID. For example:

```
AKIAIOSFODNN7EXAMPLE
```

- `origin-id` is the name that you gave when creating the configuration. For 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. 12).

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

- HLS: `https://bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com/v1/master/hashed-account-id/origin-id/master.m3u8`
- DASH: `https://bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com/v1/dash/hashed-account-id/origin-id/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. 11) must be defined in the manifest request from the player.

Example

If 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. 41).

(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 breaks 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. 47).

Topics

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 ad breaks. The service inserts the default audio from the ad and replicates it across your audio tracks during ad breaks.

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 breaks.

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 for live streaming and video on demand (VOD). Ad markers such as `SCTE-IN/OUT` and `CUE-IN/OUT` indicate ad breaks. The duration of the ad breaks is set in the `EXT-X-CUE-OUT` tag or the `EXT-X-DATERANGE Duration` tag. When MediaTailor encounters an ad break, 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 break, 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. 47).

When AWS Elemental MediaTailor stitches in the ads that are specified in the VAST response, it checks if the ads have already been transcoded. If an ad has been transcoded, MediaTailor uses the ad in the

ad break. 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, 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.

Topics

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 break 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
```

HLS Manifest Tag Handling

AWS Elemental MediaTailor outputs all unknown and custom tags into the personalized output manifest except for `EXT-X-CUE- OUT/IN` tags. This section describes how MediaTailor handles those tags.

AWS Elemental MediaTailor requires HLS `EXT-X-VERSION 3` or later as the input manifest.

Topics

`EXT-X-CUE` Tags

To identify ad creative boundaries, AWS Elemental MediaTailor converts `EXT-X-CUE-OUT`, `EXT-X-CUE-OUT-CONT`, and `EXT-X-CUE-IN` tags from the input manifest to `EXT-X-DISCONTINUITY` tags in the output manifest. MediaTailor inserts an `EXT-X-DISCONTINUITY` tag at the start and end of every ad, including the following boundaries:

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

`EXT-X-KEY` Value

When the origin server enables encryption or digital rights management (DRM) on the content stream, the manifest includes `EXT-X-KEY` tags. Ads aren't encrypted, so AWS Elemental MediaTailor sets the `EXT-X-KEY` tag to `NONE` for ad breaks. When playback returns to the content stream, MediaTailor re-enables the `EXT-X-KEY` tag.

DASH .mpd Manifests

AWS Elemental MediaTailor supports `.mpd` DASH manifests for live streaming as follows:

- Supports multi-period DASH-compliant manifests, including those from MediaPackage.
- Supports manifests for live streaming only. MediaTailor doesn't currently support VOD.
- Follows the guidelines for DASH dynamic profile.
- Requires at least one `Period` element with a `start` attribute.
- Supports 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.
- Supports segment templates that have segment timelines.
- For published manifests, 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 break periods the same `AdaptationSet` and `Representation` settings as the content stream periods. MediaTailor uses these settings to transcode the ads to match the content stream, for smooth switching between the two.

Topics

DASH Ad Markers

A `Period` in a DASH manifest is eligible for ad replacement by MediaTailor when the first event in its event stream has splice insert or time signal cue out markers. You can provide the markers in clear XML or in a base64-encoded binary:

- **Clear XML** – the event stream `schemeIdUri` must be set to `urn:scte:scte35:2013:xml`, and the first 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>
   </EventStream>
```

- `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>
    </EventStream>
```

  With time signal markers, MediaTailor only uses the duration settings and ignores all other settings.

- **Base64-encoded binary** – the event stream `schemeIdUri` must be set to `urn:scte:scte35:2014:xml+bin` and the first event must have `scte35:Signal` `scte35:Binary` that contains a base64-encoded binary. The decoded binary must provide the same set of information in its `splice_info_section` 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. With this example period, AWS Elemental MediaTailor uses only the first `Event` element and disregards the second `Event` element:

```
  <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>VGVzdCBzdHJpbmcgZm9yIGVuY29kaW5nIHRvIEJhc2U2NCBlbmNvZGVkIGJpbmFyeS4=</
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>
    <EventStream>
```

MediaTailor considers only the first `Event` in an event stream to determine ad replacement markers. It ignores any additional `Event` markers in the stream.

DASH Ad Avail Duration

During playback, when AWS Elemental MediaTailor encounters a period that is marked for ad replacement, it replaces some or all of the period with ads. MediaTailor starts ad replacement at the beginning of the period and includes ads as follows:

- If the period specifies a duration for the event, AWS Elemental MediaTailor includes as many ads as it can into the duration without going over.
- If no duration is provided, MediaTailor includes ads until it reaches an end-of-period indicator. MediaTailor doesn't play ads past the end of the period and, when it encounters a period end, truncates the ad instead of going over.

How AWS Elemental MediaTailor looks for the period 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 a period that is configured with an `Event 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 a period designated for ad replacement with no duration specified. The `Event` element 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 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.

DASH Splice Insert Example

DASH origin MPD example for splice insert

The following example from an MPD manifest shows an ad-break `Period` 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 MPD example for splice insert

AWS Elemental MediaTailor personalizes the ad-break periods 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-break period 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 Time Signal Example

DASH origin MPD example for time signal

The following example from an MPD manifest shows an ad-break `Period` in a manifest received by DASH from the content origin. This 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 MPD example for time signal

AWS Elemental MediaTailor personalizes the ad-break periods 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-break period after 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 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 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 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. 31).
  - 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 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 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 break.

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. 34).

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.

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. 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 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.
5. For **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. 11), 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.

6. For **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. 50).

   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.

7. (Optional) The **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 (p. 34).

8. (Optional) The **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:

   Example HLS

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

   Example DASH

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

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

   For more information about integrating MediaTailor with a CDN, see CDN Integration (p. 34).

9. (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. 29).

10. Choose **Create configuration**.

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

11. (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. 34).

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. 31).
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**.

Integrating with AWS Elemental MediaTailor

This section describes optional integrations with AWS Elemental MediaTailor that you can perform to optimize your manifest personalization experience.

Topics

CDN Integration

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.

  HLS 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.

DASH 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 HLS

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

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

  Example DASH

  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, for 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 manifests: `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 manifests: `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 manifests: `https://<playback-endpoint>/v1/dash/<hashed-account-id>/<session-id>/<assetName>.mpd`

  Example

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

- Ad reporting requests 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 Examples

- 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. 52).

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 break.

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. 41).

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. 43).

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 break 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 break. 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. 50) and Creating a Configuration (p. 31).

- Use client-side reporting. AWS Elemental MediaTailor supports VPAID through our client-side reporting API. For more information, see Client-side Reporting (p. 52).

  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 breaks, 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 break 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 break 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. 43).
- **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. 44).

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. 51).

   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:

Example

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

AWS Elemental MediaTailor generates data about each playback session. It uses this data when making a request to the ADS, to provide values for `session` and `avail` query parameters in the template ADS request URL. You can also concatenate multiple variables to create a value.

You can use the following variables in the template ADS request URL:

- **`[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 break in those tags, MediaTailor uses the default.
  - For DASH, MediaTailor obtains the duration value from the period's 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. 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
  ```

- **`[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.
- **`[event_id]`** – the value parsed from the SCTE-35 field `splice_event_id`, as a long number. MediaTailor uses this value to designate linear ad break numbers or to populate ad server query strings, like ad pod positions.
- **`[avail_num]`** – the value parsed from the SCTE-35 field `avail_num`. MediaTailor can use this value to designate linear ad break numbers.

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 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 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 breaks 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 `XX-OUT/XX-IN` Markers

Although `CUE-OUT/IN` (or `SCTE-OUT/IN`) markers are the preferred way of signaling ad breaks 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 breaks based on the response:

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

  Tip
  If you want to create mid-roll breaks 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.

XX-OUT/XX-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, for 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 cue-out and cue-in markers as closely as possible. The cue out can optionally indicate the duration of the ad break. Every cue-out indicator must have a matching cue-in indicator in the live workflow.

AWS Elemental MediaTailor performs ad replacement for live content for HLS and DASH, with cue-out, cue-in, and duration specified as follows:

- For HLS, cue out and cue in are specified with XX-OUT and XX-IN markers. The duration is specified in the `CUE-OUT`, or `SCTE-OUT`, marker.
- For DASH, the cue out is specified by the `Period` start and the cue in is specified by the end of the period. The duration is specified in the `duration` of the first `Event` in the period.

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 break.

Examples

- If the ad break 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 break 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 breaks. 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 break 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 breaks 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. 52). For information about VPAID, see the section called “VPAID” (p. 39).

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 break window
- If an ad isn't available

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

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: the service sends reports to the ad tracking URL directly when the player requests an ad URL from the manifest. 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 ads are played back, MediaTailor sends beacons to the ad server to report how much of the ad is viewed. Beacons track the start of an ad, ad progression in quartiles (first, midpoint, and third), and when an ad is viewed to completion.

To perform server-side ad reporting

1. From the player, initialize a new MediaTailor playback session using a request in one of the following formats, according to your protocol:

   - HLS format

     ```
     GET <mediatailorURL>/v1/master/<hashed-account-ID>/<originID>/<assetID>?ads.<key-value-pairs>
     ```

   - DASH format

     ```
     GET <mediatailorURL>/v1/dash/<hashed-account-ID>/<originID>/<assetID>?ads.<key-value-pairs>
     ```

   The `<key-value-pairs>` are the targeting parameters for ad tracking. For information about adding parameters to the request, see Player Data (p. 44).
2. AWS Elemental MediaTailor responds to the request with the manifest URL. The manifest includes URLs for the media manifests. Links for ad segment requests are embedded in the media manifests.
3. When the player requests playback from an ad segment URL (`/v1/segment` path), AWS Elemental MediaTailor sends the appropriate beacon (start, complete, and quartiles) to the ad server through the ad tracking URLs. At the same time, the service issues a redirect to the actual *.ts ad segment either in the Amazon CloudFront distribution where MediaTailor stores transcoded ads, or in the content distribution network (CDN) where you have cached the ad.

   MediaTailor sends a beacon each time a player makes a request to the `/v1/segment` URL. If the player has to make multiple requests to the same URL (in conditions such as network degradation), the service also sends multiple beacons. To avoid this duplication, use a CDN in front of MediaTailor

to cache the `/v1/segment` URL path (as described in Integrating AWS Elemental MediaTailor and a CDN (p. 34)), or consider client-side reporting (as described in Client-side Reporting (p. 52)).

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 trick play for VOD (players display visual feedback during fast forward and rewind). It also enables other advanced playback behavior during ad breaks that require player development (like no skip-forward and countdown timers on ad breaks).

Use client-side reporting for VPAID functionality. For more information, see VPAID Handling (p. 39). The client-side reporting response includes additional metadata about the VPAID creative.

To perform client-side ad reporting

1. From the player, initialize a new AWS Elemental MediaTailor playback session using a request like the following:

```
POST <mediatailorURL>/v1/session/<hashed-account-ID>/<originID>/<assetID>
    {
        "adsParams": {
           "param1": "value1",
           "param2": "value2",
           "param3": "value3"
       }
    }
```

   Note the following about the message body JSON:

   - The `adsParams` are parameter specifications that MediaTailor uses in the request to the ADS. In the MediaTailor configuration, you define these parameters as `[player_params.param]` in the ADS template URL, as described in Player Data (p. 44).
   - Any other query parameters that you provide are forwarded by MediaTailor to your origin server.
2. 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

     HLS example:

     ```
     /v1/master/<hashed-account-id>/<originID>/<assetID>?aws.sessionID=<session>
     ```

     DASH example:

     ```
     /v1/dash/<hashed-account-id>/<originID>/<assetID>?aws.sessionID=<session>
     ```

   - Tracking – used to poll for upcoming ad breaks

     Example:

     ```
     /v1/tracking/<hashed-account-id>/<originID>/<assetID>/<session>
     ```

   To construct the full manifest and tracking URLs, prefix the relative URLs with *<mediatailorURL>*.

3. The player should periodically poll the tracking URL. 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 breaks. 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 following values can be included in the response:

   - `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 along to the player.
   - `apiFramework`: Set to "`VPAID`". Tells the player this is a VPAID ad.
   - `availId`: For HLS, the sequence number associated with the start of the ad break. For DASH, the period ID of the avail, which is usually the period ID of the content that is to be replaced with an ad.
   - `beaconUrls`: Where each beacon should be sent.
   - `bitrate`: Bitrate of the video asset. This is not typically included for an executable asset.
   - `delivery`: Either `progressive` or `streaming`, depending on the protocol.
   - `duration`: Length in ISO 8601 seconds format. The response includes durations for the entire ad break and for each ad and beacon (though beacon durations are always zero). For VPAID Handling (p. 39), 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 break 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`: Assets that the player needs to know about.
   - `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 break 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 break and for each ad and beacon.
   - `apiFramework`: Set to "`VPAID`". Tells the player that this is a VPAID ad.
   - `adParameters`: String of ad parameters from VAST VPAID, which AWS Elemental MediaTailor passes along to the player.
   - `mediaFilesList`: Assets that the player needs to know about.
   - `mediaFileUri`: URI that points to either an executable or video asset. Example: "`https://myad.com/ad/ad134/vpaid.js`".
   - `delivery`: Either "`progressive`" or "`streaming`", depending on the protocol.

- `mediaType`: Typically either JavaScript or Flash for executable assets.
- `width`: Width of the video asset.
- `height`: Height of the video asset.
- `bitrate`: Bitrate of the video asset. This is not typically included for an executable asset.
- `scalable`: Indicates whether to scale the video to other dimensions.
- `maintainAspectRatio`: Indicates whether to maintain the aspect ratio while scaling.
- `mezzanine`: Specifies a mezzanine MP4 asset, if the VPAID ad includes one. Example: `"https://gcdn.2mdn.net/videoplayback/id/itag/ck2/file/file.mp4"`.

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
    }
  ]
}
```

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 more information, see the Amazon CloudWatch Logs User Guide.

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

You can monitor AWS Elemental MediaTailor using CloudWatch, which 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**).

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 in 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 fill the times available for the corresponding ad breaks. Each rate is defined as (`AdDecisionServer.Duration`)/(`Avails.Duration`). This number can be higher than 100%. For information about the simple average, see Simple Average Explanation (p. 60). For example, for a single ad break, if the ADS returns 120 seconds of ads and the ad break is 180 seconds, then the `AdDecisionServer.FillRate` for this single ad break is 67% (120/180). The best `Avails.FillRate` that MediaTailor can attain for this ad break is 67%. If your ADS returns 120 seconds of ads and the ad break is 90 seconds, then the `AdDecisionServer.FillRate` is 133% (120/90). The best `Avails.FillRate` that MediaTailor can attain for this ad break is therefore 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.

Metric	Description
	A high value for this metric might contribute to a low overall `Avails.FillRate`.
`Avails.Duration`	The total duration, in milliseconds, of all ad breaks that MediaTailor encountered in the time period that you specified.
`Avails.FilledDuration`	The total duration, in milliseconds, of all ad breaks that MediaTailor filled in the time period that you specified. This metric is calculated as (number of concurrent sessions) x (ad break duration filled).
`Avails.FillRate`	The simple average of the rate at which MediaTailor filled ad breaks. The rate is defined as (`Avails.FilledDuration`) / (`Avails.Duration`). For information about the simple average, see Simple Average Explanation (p. 60). For example, if your ADS returns 90 seconds of ads and the ad break is 120 seconds, then the `AdDecisionServer.FillRate` for this single ad is 75% (90/120). If the `Avails.FillRate` is low, look at the `AdDecisionServer.FillRate` compared to the `Avails.FillRate`. If the `AdDecisionServer.FillRate` is low (50% or lower) and `Avails.FillRate` is also low, your ADS might be returning only enough ads for half of a typical break duration. The maximum `Avails.FillRate` that MediaTailor can attain is bounded by the `AdDecisionServer.FillRate`.
`GetManifest.Errors`	The number of errors received when MediaTailor is generating manifests.
`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 Average Explanation

Simple average means that the durations aren't weighted in the `FillRate` calculation. The `FillRate` for each ad break is simply averaged together. This gives an overall view of how successful ad insertions are across all of your ad breaks, independent of how long each ad break is. To get a weighted average, calculate the sum of your **total duration** and divide by the **total filled duration** in a time period.

Example

You have the following two ad breaks:

- Ad break A: 90 seconds total duration, 45 seconds filled (50% filled)
- Ad break B: 120 seconds total duration, 120 seconds filled (100% filled)

The `FillRate` metric reported is 75% ([50% + 100%] / 2).

The actual weighted average `FillRate` is 79% ([45 seconds filled + 120 seconds filled] / [90 total seconds + 120 total seconds]).

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.

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"
}
```

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*.

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 break 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.

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.

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/.

Co	Exception Name	Meaning	What To Do
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	`UnsupportedManifestFormat`	MediaTailor encountered a problem with the session's playlist.	This affects only the individual session. Reinitialize the session. You can usually accomplish this by refreshing the page in the viewer.
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. 66).
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. 66).

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: 1111222233334444555566661234567890 12.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 1111222233334444555566661234567890 12.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:[12345678901 2123456789012] customerId:
[123456789012]"}%
```

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 2017	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.

- **API version:** 1.0

update-history-change	update-history-description	update-history-date
Tagging support (p. 64)	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. 61)	Added topic about using CloudTrail to log actions in the AWS Elemental MediaTailor API.	February 11, 2019
Added section on playback errors (p. 68)	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. 19)	Added support for providing splicing information in manifests in a base64-encoded binary, inside `<scte35:Signal>` `<scte35:Binary>` markers.	January 4, 2019
DASH time signal (p. 19)	Added support for providing splicing information in manifests inside `<scte35:TimeSignal>` markers.	December 5, 2018
DASH location support (p. 29)	Added support for the MPEG-DASH `<Location>` tag.	December 4, 2018
DASH Support (p. 16)	Added support for MPEG-DASH manifests.	November 14, 2018
Updated limits tables. (p. 66)	Updated limits for configurations and manifest size.	October 13, 2018
New and updated metrics. (p. 58)	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. 41)	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. 38)	Added information about VAST and VPAID.	March 16, 2018
CloudWatch (p. 57)	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. 34)	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*.