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.

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.

MediaTailor supports Apple HTTP Live Streaming (HLS) manifest manipulation. If you require support for MPEG-DASH, create a feature request case with AWS Support.

Topics

Are You a First-Time User of AWS Elemental MediaTailor?

If you are a first-time user of AWS Elemental MediaTailor, we recommend that you begin by reading the following sections:

- Concepts and Terminology (p. 1)
- How AWS Elemental MediaTailor Works (p. 2)
- Features of AWS Elemental MediaTailor (p. 3)
- Getting Started with AWS Elemental MediaTailor (p. 10)

Concepts and Terminology

Configuration

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

Dynamic transcoding

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

Manifest manipulation

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

VAST and VMAP

Video Ad Serving Template (VAST) and Video Multiple Ad Playlist (VMAP) are XML responses that the ADS sends to ad requests from 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. 33). For more information about how MediaTailor works with VAST, see VAST (p. 25).

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.

The general MediaTailor processing flow is as follows:

1. A player or content distribution network (CDN) such as Amazon CloudFront sends a request for live or video-on-demand (VOD) HLS content to MediaTailor. The request includes parameters from the player that include information about the viewer. Later, the ad decision server (ADS) uses these parameters to determine which ads are included in the MediaTailor response to the content request. 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. 37). For information about configuring the ad targeting parameters, see Dynamic Ad Variables in AWS Elemental MediaTailor (p. 28).
2. MediaTailor pulls the fully formed template manifest from the content origin server (such as AWS Elemental MediaPackage). This manifest includes ad markers so that MediaTailor knows where to perform an ad insertion or ad replacement.
3. Additionally, MediaTailor sends a request to the ad decision server (ADS), including the player parameters from the content request.
4. The ADS provides a VAST or VMAP response that includes the ads to be played back, based on viewer information gathered from the parameters that MediaTailor passed through, and current ad campaigns.
5. 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. 33).
6. MediaTailor provides the fully customized manifest to the requesting CDN or player.
7. As playback progresses, either MediaTailor or the video player reports how much of an ad is played. By default, MediaTailor uses server-side reporting, meaning that the service sends ad viewing reports to the ad tracking URL directly, with no input required from you. If you require more control, you can instead perform client-side ad reporting, where MediaTailor proxies the ad tracking URL to the player for it to perform ad tracking activities.

8. As the player requests ad segments throughout content playback, if the ad is not already transcoded in a format that matches the video content, MediaTailor transcodes the ad at the time of the ad segment request. If an ad is not already transcoded, the service doesn't present it for playback at the first request.

Mixed Content Requests

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

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

Manifest Response Latency

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

- Manifest processing latency – time 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

MediaTailor supports the following features:

Ad Tracking Reporting

MediaTailor offers both server-side and client-side ad view reporting:

- 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 call directly to the ad decision server (ADS) for reporting on 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. 37).

Audio

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

Content and Ad Continuity

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

Personalized Content

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

Related Services

- **Amazon CloudFront** is a global content delivery network (CDN) service that securely delivers data and videos to your viewers. Use CloudFront to deliver content with the best possible performance. For more information about CloudFront, see the Amazon CloudFront website.
- **AWS Elemental MediaPackage** is a just-in-time packaging and origination service that customizes live video assets for distribution in a format that is compatible with the device that makes the request. Use AWS Elemental MediaPackage as an origin server to prepare content and add ad markers before sending streams to MediaTailor. For more information about how MediaTailor works with origin servers, see How AWS Elemental MediaTailor Works (p. 2).
- **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. 6).

Accessing AWS Elemental MediaTailor

You can access MediaTailor using the service's console.

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

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

Pricing for AWS Elemental MediaTailor

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

Regions for AWS Elemental MediaTailor

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

Stream Requirements

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

- Uses HLS (Apple HTTP Live Streaming)
- Uses live streaming or video-on-demand (VOD)

- Is accessible on the public internet and has a public IP address
- Contains ad markers in one of the formats described in Step 2: Prepare a Stream (p. 10)

Sensitive Information

AWS Elemental MediaTailor does not require that you supply any customer data.

Do not 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, do not 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 ads in a video stream, depending on where your ad data is located, it might get copied to another AWS Region.

If you need to delete your transcoded ad assets for any reason, file a basic support ticket through the AWS Support console with the category of "service limit increase."

Setting Up AWS Elemental MediaTailor

Before you start using 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 this procedure, you will use the AWS account root user to create your first IAM user. You will 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, you should sign in with the credentials for this IAM user.

To create users with limited permissions, see Creating a Non-Admin IAM User (p. 7).

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: 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. 7).

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 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 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 MediaTailor.
5. Choose **Review policy**.
6. On the **Review policy** page, for **Name**, type **`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 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, type 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 MediaTailor, and 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 MediaTailor

1. In the navigation pane of the IAM console, choose **Users**, and then choose **Add user**.
2. For **User name**, type 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 type 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 an MediaTailor configuration that holds information about the origin server and ad decision server (ADS).

Topics

Prerequisites

Before you can use AWS Elemental MediaTailor, you need an AWS account and the appropriate permissions to access, view, and edit MediaTailor configurations. Complete the steps in Setting Up AWS Elemental MediaTailor (p. 6), and then return to this tutorial.

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 appropriately formatted HLS manifests.

Manifests must satisfy the following requirements:

- Manifests must be live or video-on-demand (VOD).
- Manifests must be accessible on the public internet.
- 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 in AWS Elemental MediaTailor (p. 33).

- The master playlists in the 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`.

When the stream is configured, note the URL to its master playlist. You need the URL when you create the configuration in MediaTailor, in a later step of this procedure.

Step 3: Configure ADS Request URL and Query Parameters

To determine the query parameters that the ad decision server (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
- MediaTailor-generated values (denoted by `session` or `avail` query parameters)
- Player-generated values, obtained from the client application (denoted by `player_params.` query parameters)

Example

```
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 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 have corresponding key-value pairs for all `player_params.` query parameters in the ADS request URL. For more information about configuring key-value pairs in the request to MediaTailor, see Dynamic Ad Variables in AWS Elemental MediaTailor (p. 28).

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

Testing Purposes

You can use a static VAST response from your ADS for testing purposes. Ideally, the VAST response returns a mezzanine quality *.mp4 rendition that MediaTailor can transcode upon receipt. 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 MediaTailor configuration holds mapping information for the origin server and ad decision server (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. For **Configuration name**, type 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**, type the URL prefix for the master playlist for this stream, minus the asset ID. For example, if the master playlist URL is `http://origin-server.com/a/master.m3u8`, you would type `http://origin-server.com/a/`. Alternatively, you can type 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. The maximum length is 512 characters.

   Note
   If your content origin uses HTTPS, its certificate must be from a well-known certificate authority (it cannot be a self-signed certificate). Otherwise, MediaTailor fails to connect to the content origin and can't serve manifests in response to player requests.
5. For **Ad decision server**, type 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 25000 characters.

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

   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 following format:

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

Where:

- *playback-endpoint* is the unique playback endpoint that MediaTailor generated when the configuration was created:

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

- *hashed-account-id* is your AWS account ID:

```
111122223333
```

- *origin-id* is the name that you gave when creating the configuration:

```
myOrigin
```

- *assetID.m3u8* is the name of the master playlist from the test stream, excluding the URL path elements that you gave when adding the origin server to the MediaTailor configuration.

Using the values from the preceding examples, the full URL is the following:

```
https://bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com/v1/
master/111122223333/myOrigin/assetID.m3u8
```

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

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

- Test the stream in your own player environment.

Step 6: Send the Playback Request to MediaTailor

Configure the downstream player or CDN to send playback requests to the configuration's playback endpoint provided from 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.`. MediaTailor passes any parameters that aren't preceded with `ads.` to the origin server instead of the ADS.

The player request URL is some variation of this:

```
https://bdaaeb4bd9114c088964e4063f849065.mediatailor.us-east-1.amazonaws.com/v1/
master/111122223333/myOrigin/assetId.m3u8?ads.cust_params=viewerinfo
```

When 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. 28).

(Optional) Step 7: Monitor MediaTailor Activity

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

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

To allow 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**, type 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 choose **Next: Review**:
   - **CloudWatchLogsFullAccess** to provide full access to Amazon CloudWatch Logs.
   - **CloudWatchFullAccess** to provide full access to Amazon CloudWatch.
8. For **Role name**, type **`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 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 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, type **`Delete`**, and then choose **Delete** again.

   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 each viewing session. The following sections describe the expected general behaviors of MediaTailor manifest handling. For information about ad handling and insertion, see Ad Behavior in AWS Elemental MediaTailor (p. 33).

Topics

Alternate Audio and Subtitles

AWS Elemental MediaTailor supports input and output of multiple audio and WebVTT subtitle tracks. To learn how MediaTailor handles these tracks, see the following sections.

Topics

Alternate Audio Expected Behavior

If your content contains alternate audio, 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 always inserts the default audio from the ad and replicates it across your audio tracks during ad breaks.

The audio sample rate must be from 16 to 320 kHz for ad transcoding to succeed.

Subtitles Expected Behavior

Ad playback does not include subtitles. Instead, MediaTailor inserts blank offsets for the webVTT sidecar files during ad breaks.

HLS .m3u8 Manifests

AWS Elemental MediaTailor supports HLS manifests (*.m3u8) 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 determined by the value in the `EXT-X-CUE-OUT` tag or by `EXT-X-DATERANGE Duration`. 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, MediaTailor displays the underlying content stream or the configured slate for the remainder of the ad break. For more information about HLS ad behavior based on content type (live or VOD), see Ad Behavior in AWS Elemental MediaTailor (p. 33).

When MediaTailor stitches in ads, it first checks to see if the ads returned in the VAST response of the ad decision server (ADS) have 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 and stores it for future use. If there are multiple ads in the VAST response, MediaTailor evaluates them sequentially and attempts to fill in subsequent ad creatives if the ads 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 playlist as input to 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 playlist as input to MediaTailor. Note the `EXT-X-CUE-OUT` and `EXT-X-CUE-IN` tags describing 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

MediaTailor outputs all unknown and custom tags into the personalized output manifest with the exception of `EXT-X-CUE- OUT/IN` tags, whose handling is described below.

To work properly, MediaTailor requires HLS `EXT-X-VERSION 3` or higher as the input manifest.

Topics

EXT-X-CUE Tags

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 to identify discrete ad creative boundaries. 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

If the origin server has enabled encryption or digital rights management (DRM) on the content stream, the manifest includes `EXT-X-KEY` tags. Because ads aren't encrypted, 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.

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 what the playback defaults to if an ad isn't available or doesn't fill the entire ad break.

If you are using a content distribution network (CDN) with MediaTailor, you have to 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 MediaTailor and a CDN (p. 22).

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**, type 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**, type the URL prefix for the master playlist for this stream, minus the asset ID. For example, if the master playlist URL is `http://origin-server.com/a/master.m3u8`, you would type `http://origin-server.com/a/`. Alternatively, you can type a shorter prefix such as `http://origin-erver.com` but the `/a/` must be included 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 cannot be a self-signed certificate). Otherwise, MediaTailor fails to connect to the content origin and can't serve manifests in response to player requests.
5. For **Ad decision server**, type 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 25000 characters.

   Note
   If your ADS uses HTTPS, its certificate must be from a well-known certificate authority (it cannot be a self-signed certificate). The same also applies to mezzanine ad URLs returned by the ADS. Otherwise, MediaTailor can't retrieve and stitch ads into the manifests from the content origin.
6. For **Slate ad**, type the URL for a high-quality MP4 asset to transcode and use to fill in time that's not used by ads. 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. 36).

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

7. (Optional) The **CDN content segment prefix** enables 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**, type the CDN prefix path.

   For more information about integrating MediaTailor with a CDN, see CDN Integration (p. 22).
8. (Optional) The **CDN ad segment prefix** enables 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:

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

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

   For more information about integrating MediaTailor with a CDN, see CDN Integration (p. 22).
9. Choose **Create configuration**.

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

    For information about setting up a CDN for manifest and reporting requests, see Integrating MediaTailor and a CDN (p. 22).

Viewing a Configuration

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, MediaTailor displays the name of the configuration, playback endpoints, and relevant access URLs.

Editing a Configuration

Edit a configuration to update the origin server and ad decision server (ADS) mapping, or change how 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. Note that you can't edit the configuration name. For information about configuration attributes, see Creating a Configuration (p. 19).
4. Choose **Save**.

Deleting a Configuration

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 radio button, and then choose **Delete**.
3. In the **Delete** confirmation box, type **`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 master manifest from the CDN with MediaTailor as the manifest origin. Personalized manifest requests are proxied through the CDN, and the CDN forwards the requests 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 and consequently to the requesting player.
3. The player requests segments from the URLs that are provided in the master manifest.
4. The CDN translates the segment URLs and forwards content segment requests to the origin server and 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 MediaTailor and the CDN to perform this flow.

Topics

Integrating MediaTailor and a CDN

The following steps show how to integrate 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 MediaTailor, as follows:

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

For example, the CDN routes 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.

- Create one behavior that routes *ad segment* requests to the internal Amazon CloudFront distribution where MediaTailor stores transcoded ads, based on a rule that includes a phrase to differentiate ad segment requests from content segment requests.

  These are the default Amazon CloudFront distributions that MediaTailor uses for storing ads:

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

  This step is optional because MediaTailor provides a default CDN configuration for ad serving.

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

Create an MediaTailor configuration that maps the domains of the CDN routing behaviors to the origin server and to the location where the ads are stored. Type the domain names in the configuration as follows:

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

  For example,

  - If the full content file path is `http://origin.com/contentpath/subdir/content.ts`,
  - then the **Video content source** in the MediaTailor configuration is `http://origin.com/contentpath/`,
  - and the **CDN content segment prefix** is `https://CDN_Hostname/`,
  - then the content segment advertised in the master manifest that MediaTailor serves is `https://CDN_Hostname/subdir/content.ts`.
- For **CDN ad segment prefix**, type the name of the CDN behavior that you created to route ad requests through your CDN. In the playlist 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 enables additional functionality in your workflow.

For manifests, referencing a CDN in front of `/v1/master` (in master playlist requests) or `/v1/manifest` (for manifest playlist requests) 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.

For reporting, referencing a CDN in front of `/v1/segment` in ad segment requests helps prevent 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 MediaTailor configuration endpoint based on rules that differentiate requests for master manifests, media playlists, and reporting. Requests follow these formats:

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

  Example

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

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

  Example

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

- Ad reporting requests for server-side reporting: `https://<playback-endpoint>/v1/segment/<origin-id>/<session-id>/<playlistNum>/<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 MediaTailor configuration endpoint based on a rule that includes a phrase to differentiate the manifest request from segment requests.

For example, player requests to `https://CDN_Hostname/some/path/asset.m3u8` are routed to the MediaTailor path `https://mediatailor.us-west-2.amazonaws.com/v1/session/configuration/endpoint` based on the keyword ***.m3u8** in the request.

VAST

This topic covers the use of the Interactive Advertising Bureau (IAB) Video Ad Serving Template (VAST), Video Player Ad-Serving Interface Definition (VPAID), and 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 on client-side reporting, see Client-side Reporting (p. 38).

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 these tags are well-formed.

Make sure that your ad server's VAST response contains 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 fill for the break.

VAST 3.0 introduced support for ad pods, which is the delivery of a set of sequential linear ads. With AWS Elemental MediaTailor 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 ad decision servers, and tries to insert the next ad in the pod. In this way, AWS Elemental 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. 28).

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. 29).

Ad Calls

MediaTailor calls your VAST ads URL as defined in your configuration, substituting 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 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 does not currently support pre-fetching VAST responses.

Creative Handling

When AWS Elemental MediaTailor receives the ADS VAST response, for each creative it identifies the highest bit rate `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 bit rates and resolutions. For best results, make sure that your highest bit rate media file is a high-quality MP4 asset with valid manifest presets. When manifest presets are not valid, the transcode jobs fail, resulting in no ad shown. Examples of presets that are not 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, AWS Elemental 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. Do not 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, AWS Elemental MediaTailor uses the stored content, rather than reprocessing the incoming content. If you reuse a creative ID, AWS Elemental MediaTailor uses the older, stored ad and does not play your new ad.

VPAID Handling

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

Follow these guidelines to use VPAID:

- 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 that is used on the client side to run the interactive ads. If you do not have a slate configured, when a VPAID ad appears, AWS Elemental MediaTailor logs an error in CloudWatch. It inserts MP4 ads normally no matter what. For more information, see Slate Management (p. 36) and Creating a Configuration (p. 19).
- Use client-side reporting. AWS Elemental MediaTailor supports VPAID through our client-side reporting API. For more information, see Client-side Reporting (p. 38).

  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 only available 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. In your ADS request configuration, you specify the query parameters to use to convey the viewing session 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. 29).
- **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. 30).

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 `user` provides player data:

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

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. 37).

   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 parameters for the ADS with `ads` and omit the `ads` prefix in the parameters that you want MediaTailor to send to the origin server:

Example

```
GET master.m3u8?ads.userID=xyzuser&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:

```
POST master.m3u8
    {
        "adsParams": {
           "userID": "xyzuser"
       }
       "auth_token": "kjhdsaf7gh"
    }
```

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

The following examples show calls 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 user ID, and calls the origin server with the player's authorization token:

```
https://my.ads.server.com/path?correlation=896976764&user=xyzuser
```

```
https://my.origin.server.com/master.m3u8?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 that is being requested. MediaTailor obtains the duration value from the input manifest's `#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 a default value of 300,000 ms.
- **`[session.avail_duration_secs]`** – the duration in seconds of the ad availability slot that is being requested. MediaTailor obtains the duration value from the input manifest's `#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 a default value of 300 seconds.
- **`[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 during a session have the same value for this field, 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. This variable is set to the value of the `Referer` header that the player uses in its request to MediaTailor. If the player doesn't include this header, the **[session.referer]** value is empty. 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.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 AWS Elemental MediaTailor and you need to pass that data in the ADS request, then include `[player_params.user_id]` in the ADS URL configuration.

This functionality 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 a session initialization request from the player for a session with server-side reporting, 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 request for server-side ad-tracking reporting - using URL-encoded pairs

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

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

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

For server-side reporting, MediaTailor decodes the parameters when the player request is received. For client-side reporting, it does not 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

AWS Elemental MediaTailor can perform ad replacement (replace content segments with ad content) or ad insertion (insert ad content where segments don't currently exist). The ad behavior depends on the type of content (VOD or live), and how the origin server configured the ad breaks. Additionally, AWS Elemental MediaTailor uses configured slates to fill gaps in ads and manage VPAID ad handling.

Generally, the ad flow goes like this:

1. The player requests a master manifest from MediaTailor.
2. MediaTailor requests a VAST (or VMAP) response from the ad decision server (ADS) and master manifest from the origin server.
3. MediaTailor stitches ads into the master manifest based on the response from the ADS.

   The following sections describe the logic that MediaTailor uses when stitching ads into a manifest.

Topics

VOD Content Ad Behavior

AWS Elemental 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 ad decision server (ADS) sends VMAP responses.

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

Topics

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 ad decision server (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 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
```

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 cannot 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 above 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, with the total time between XX-OUT and XX-IN markers preserved as closely as possible. Note the following about live ad insertion:

- MediaTailor always prioritizes the content stream over ad content. If MediaTailor encounters an early CUE-IN before the ad break time has elapsed, the ad might be truncated.
- If there aren't enough ads in the VAST response to fill the ad break in the manifest, MediaTailor plays the underlying stream (or the ad slate if one was provided in the ADS and origin server configuration).
- If the fragment duration for individual ads exceeds the ad break in the manifest, MediaTailor fits as many complete ads as it can. When it can't fit any more complete ads, MediaTailor plays the slate or underlying stream.

  If the ad break is for 70 seconds but the VAST response includes two ads, each of which is 40 seconds, MediaTailor plays one ad for a total of 40 seconds and then displays the configured ad slate or underlying content stream for the remaining 30 seconds of the ad break.

Ad behavior is further refined by the length of the CUE-OUT duration, as described in the following sections.

XX-OUT Duration Greater Than Zero

If the CUE-OUT (or SCTE-OUT) duration is greater than zero, MediaTailor replaces as many ads that fit in the ad break without truncation:

- If the VAST response includes a single ad and the ad break duration is less than the ad creative duration, MediaTailor doesn't splice any ads into the content stream. Instead, the service displays the configured ad slate or underlying content stream for the duration of the break.
- If the CUE-IN is presented earlier than expected, MediaTailor honors the CUE-IN and returns to the content stream, possibly cutting off some of the ad.
- If the CUE-IN is not encountered by the time the CUE-OUT duration is reached, MediaTailor ends the ad break and the stream returns to the content stream.

XX-OUT Duration Equal to Zero

If the CUE-OUT (or SCTE-OUT) duration is zero, MediaTailor splices in all ads from the ADS response until it encounters a CUE-IN marker. No CUE-IN markers in a live scenario is an error state that requires attention.

Slate Management

In AWS Elemental MediaTailor, you can configure a URL to an MP4 slate, to be used to fill gaps in media content. AWS Elemental MediaTailor inserts the slate into unfilled and partially filled ad breaks. AWS Elemental 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. AWS Elemental MediaTailor inserts the slate for the duration of the VPAID ad, though in certain cases, to accommodate user interactivity, this duration may 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 AWS Elemental MediaTailor returns. For information about client-side reporting, see Client-side Reporting (p. 38). For information about VPAID, see VPAID Handling (p. 26).

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.

MediaTailor shows the slate for the following situations:

- To fill in time that's not fully used by an ad replacement
- If the ad isn't available
- 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

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 playlist 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 AWS Elemental MediaTailor playback session using a request in the following format:

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

   where `<key-value-pairs>` are the targeting parameters for ad tracking. For information about adding parameters to the request, see Player Data (p. 30).
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), 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 MediaTailor and a CDN (p. 22)), or consider client-side reporting (as described in Client-side Reporting (p. 38)).

Client-side Reporting

With client-side reporting, 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) and other advanced playback behavior during ad breaks that requires 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. 26). 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 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"
       }
    }
```

   In 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. 30).
   - 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

     Example:

     ```
     /v1/master/<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.

   - `adID`: HLS sequence number associated with the beginning of this ad.
   - `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. 26), the duration conveyed is the MP4 slate duration. This duration is typically 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 to entirely replace 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).
- `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.
- `beaconUrls`: where each beacon is sent.
- `eventId`: HLS sequence number associated with the beacon.
- `eventType`: type of beacon.
- `availId`: HLS sequence number associated with the start of the ad break.
- `apiFramework`: Set to "`VPAID`". Tells the player 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`: Bit rate 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`".

Example responses:

```
{
  "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 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). 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**, type your AWS account ID.
5. Select **Require external ID** and type `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**, type `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 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.

To view metrics using the CloudWatch console

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

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 ad decision server (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 ad decision server (ADS) fill the time 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. 46). 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. A high value for this metric might contribute to a low overall `Avails.FillRate`.

Metric	Description
`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. 46). 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 MediaTailor data using the following dimensions.

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

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 cannot 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 (Gateway Timeout) response errors.
Manifest size	2	The maximum size, in MB, of any playback manifest, whether in input or output. To ensure that you stay under the limit, use gzip to compress your input manifests into MediaTailor.
Session expiration	10 times the manifest duration	The maximum amount of time that MediaTailor allows a session to remain inactive before ending the session. Session activity can be a player request or an advance by the origin server. When the session expires, MediaTailor returns an HTTP 400 (Bad Request) response error.

AWS Elemental MediaTailor Resources

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

Resource	Description
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
Updated limits tables. (p. 48)	Updated limits for configurations and manifest size.	October 13, 2018
New and updated metrics. (p. 44)	Added metrics for ad decision server (ADS) and origin timeouts and updated the ADS and origin errors definiitions to include timed-out responses.	October 13, 2018
Better documentation coverage for server-side and client-side ad insertion use cases. (p. 28)	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 Region (p. 4)	Supported regions now include Oregon.	July 25, 2018
VAST/VPAID (p. 25)	Added information about VAST and VPAID.	March 16, 2018
CloudWatch (p. 43)	Added information about available CloudWatch metrics, namespaces, and dimensions.	March 16, 2018
New Regions (p. 4)	Supported regions now include Singapore, Sydney, and Tokyo.	February 8, 2018
Default Amazon CloudFront distribution paths (p. 22)	Added the list of paths for the Amazon CloudFront distribution where MediaTailor stores ads.	February 6, 2018
IAM policy information (p. 6)	Added IAM policy information specific to AWS Elemental MediaTailor. Added instructions for creating non-admin roles with limited permissions.	January 3, 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*.