aws

User Guide

AWS Elemental MediaTailor

AWS Elemental MediaTailor: User Guide

Table of Contents

What is AWS Elemental MediaTailor? 1
MediaTailor concepts 1
Ad insertion concepts 1
Channel assembly concepts 2
How MediaTailor ad insertion works 3
Related services 4
Accessing MediaTailor 4
Pricing 5
Regions 5
Setting up 6
Sign up for an AWS account 6
Create a user with administrative access 6
Getting started with MediaTailor 9
Getting started with MediaTailor ad insertion 9
Prerequisites 10
Step 1: Access MediaTailor 10
Step 2: Prepare a stream 10
Step 3: Configure ADS request URL and query parameters 12
Step 4: Create a configuration 13
Step 5: Test the configuration 15
Step 6: Send the playback request to MediaTailor 16
(Optional) step 7: Monitor activity 17
Step 8: Clean up 18
Getting started with MediaTailor channel assembly 19
Prerequisites 20
Step 1: Create a source location 20
Step 2: Add VOD sources to your source location 21
Step 3: Create a channel 22
Step 4: Add programs to your channel's schedule 23
Step 5 (*optional*): Use MediaTailor to insert personalized ads into your stream 25
Step 6: Start your channel 26
Step 7: Test your channel 26
Step 8: Clean up 26
Inserting ads 28

Supported audio and video codecs ... 29
Understanding ad insertion behavior ... 29
Ad stitching behavior for VOD ... 29
Live ad stitching behavior ... 32
Understanding server-guided ad insertion ... 33
Enable in the playback configuration ... 34
Create a server-guided session ... 34
Requirements for ad server integrations ... 35
VAST requirements ... 36
VPAID requirements ... 38
Working with playback configurations ... 39
Creating a configuration ... 39
Viewing a configuration ... 46
Editing a configuration ... 46
Deleting a configuration ... 47
Integrating a content source ... 47
Input source requirements ... 48
Integrating an HLS source ... 48
Integrating an MPEG-DASH source ... 57
Securing origin interactions with SigV4 ... 118
Integrating with Google Ad Manager ... 122
Server-side integration ... 122
Client-side integration ... 123
Using a CDN ... 124
Integrating a CDN ... 125
How MediaTailor handles BaseURLs for DASH ... 129
CDN best practices ... 129
Customizing ad break behavior with ad suppression ... 130
Configuring ad break suppression ... 130
Inserting bumpers ... 144
Configuring bumpers ... 144
Inserting pre-roll ads ... 145
Inserting slate ... 146
Configuring the slate ... 147
VPAID requirements ... 147
Prefetching ads ... 147

How prefetching works ... 148
Creating prefetch schedules ... 149
Deleting prefetch schedules ... 152
Preconditioned ads ... 153
Preconditioned ads requirements ... 154
Preconditioned ads workflow ... 157
Using dynamic ad variables ... 158
Passing parameters to the ADS ... 158
Using domain variables ... 162
Using session variables ... 165
Using player variables ... 177
Passing query parameters to the manifest ... 179
HLS implicit session initialization ... 180
DASH implicit session initialization ... 181
HLS and DASH explicit session initialization ... 183
Reporting ad tracking data ... 183
Server-side tracking ... 184
Client-side tracking ... 185
Overlay ads ... 279
Prerequisites for using overlay ads ... 280
Getting started ... 281
Logging and metrics ... 294
Billing for overlay ads in MediaTailor ... 296
Ad ID decoration ... 296
Session State ... 297
Manifests and ad metadata insertion ... 299
Ad Decision Server (ADS) interactions ... 323
Client-side tracking API ... 324
Creating linear assembled streams ... 326
Working with source locations ... 326
Creating a source location ... 327
Configuring authentication for your source location ... 329
Working with VOD sources ... 338
Working with live sources ... 342
Using package configurations ... 346
Manifest caching ... 347

Working with channels ... 347
Create a channel ... 347
Using source groups with your channel's outputs ... 350
Delete a channel ... 351
Adding a program ... 351
Creating a program ... 351
Defining audience cohorts and alternate content ... 358
Generating audience-specific manifests ... 361
Insert ads and ad breaks ... 361
Set up ad insertion ... 361
SCTE-35 messages for ad breaks ... 363
Enable time-shifted viewing ... 369
Time-shifting parameters for manifest requests ... 370
Using time-shifted viewing with CDNs ... 372
Troubleshooting playback errors ... 374
Client errors ... 375
Server errors ... 377
Examples ... 378
Security ... 380
Data protection ... 381
Data encryption ... 382
Identity and Access Management ... 382
Audience ... 383
Authenticating with identities ... 383
Managing access using policies ... 387
How AWS Elemental MediaTailor works with IAM ... 389
Identity-based policy examples ... 395
Resource-based policy examples ... 398
AWS managed policies ... 399
Using service-linked roles ... 401
Troubleshooting identity and access ... 404
Compliance validation ... 406
Resilience ... 407
Infrastructure security ... 407
Cross-service confused deputy prevention ... 408
Logging and monitoring ... 409

CloudWatch Alarms ... 410
CloudTrail logs ... 410
AWS Trusted Advisor ... 410
Monitoring and tagging ... 411
Viewing logs ... 412
ADS logs ... 412
Manifest logs ... 461
Transcode logs ... 465
Using vended logs ... 468
Writing logs to CloudWatch Logs ... 473
Controlling the volume of ad insertion session logs ... 483
Filtering logs and events ... 486
Generating debug logs ... 488
Monitoring with CloudWatch metrics ... 493
AWS Elemental MediaTailor CloudWatch metrics ... 493
AWS Elemental MediaTailor CloudWatch dimensions ... 502
Using metrics to diagnose stale manifests ... 503
Recording API calls ... 505
AWS Elemental MediaTailor information in CloudTrail ... 506
Understanding AWS Elemental MediaTailor log file entries ... 507
Receiving Channel Assembly alerts ... 509
Viewing alerts ... 514
Handling alerts ... 515
Tagging resources ... 515
Supported resources ... 515
Tag restrictions ... 516
Managing tags ... 516
Workflow monitor ... 516
Components of workflow monitor ... 518
Supported services ... 518
Configuring workflow monitor ... 519
Using workflow monitor ... 538
Quotas ... 541
Quotas on ad insertion ... 541
Quotas on channel assembly ... 545
MediaTailor resources ... 550

Document history 552

What is AWS Elemental MediaTailor?

AWS Elemental MediaTailor is a scalable ad insertion and channel assembly service that runs in the AWS Cloud. With MediaTailor, you can serve targeted ad content to viewers and create linear streams while maintaining broadcast quality in over-the-top (OTT) video applications. MediaTailor ad insertion supports Apple HTTP Live Streaming (HLS) and MPEG Dynamic Adaptive Streaming over HTTP (DASH) for video on demand (VOD) and live workflows.

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

AWS Elemental MediaTailor channel assembly is a manifest-only service that allows you to create linear streaming channels using your existing video on demand (VOD) content. MediaTailor never touches your content segments, which are served directly from your origin server. Instead, MediaTailor fetches the manifests from your origin, and uses them to assemble a live sliding manifest window that references the underlying content segments.

MediaTailor channel assembly makes it easy to monetize your channel by inserting ad breaks into your stream without having to condition it with SCTE-35 markers. You can use channel assembly with MediaTailor ad insertion, or another server-side ad insertion service.

MediaTailor concepts

Here's an overview of the concepts that are used throughout the *AWS Elemental MediaTailor User Guide*.

Ad insertion concepts

Here's an overview of the concepts that are related to ad insertion.

Ad decision server (ADS)

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

Configuration

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

Dynamic transcoding

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

Manifest manipulation

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

VAST and VMAP

Video Ad Serving Template (VAST) and Video Multiple Ad Playlist (VMAP) are XML responses that the ad decision server sends to ad requests from MediaTailor. The responses dictate what ads MediaTailor inserts in the manifest. VMAP also includes timing for ad avails. For more information about the logic behind MediaTailor ad insertion, see Understanding AWS Elemental MediaTailor ad insertion behavior. For more information about how MediaTailor works with VAST, see the section called “Requirements for ad server integrations”.

Channel assembly concepts

Here's an overview of the concepts that are related to channel assembly.

Channels

A channel assembles your source manifests into a linear stream. Each channel has one or more outputs that contain playback URLs accessed by players. The channel outputs correspond to the package configuration settings you create for your VOD sources. A channel contains a schedule, which determines when VOD sources will play in the channel's stream.

Package configuration

A packager configuration is a representation of your VOD source that contains specific packaged format characteristics. You associate your package configurations with channel outputs to

create playback streams for your VOD source's packaged formats, such as HTTP Live Streaming (HLS).

Schedule

Each channel is made up of programs that are arranged into the channel's schedule. The schedule determines what time the programs will play in the channel's linear stream.

Source locations

A source location represents the origin server where your assets are stored. It can be Amazon S3, an HTTP server, a Content Delivery Network (CDN), or a packaging infrastructure such as MediaPackage.

VOD sources

A VOD source represents a single piece of content, such as a movie or an episode of a TV show. You associate VOD sources with programs to add them to your channel's linear stream.

Audience

An audience defines a viewer cohort which can optionally have alternate content. You can define audiences on standard linear channels.

How MediaTailor ad insertion works

MediaTailor interacts between your content delivery network (CDN), origin server, and ad decision server (ADS) to stitch personalized ads into live and video on demand content.

Here's an overview of how MediaTailor ad insertion works:

1. A player or CDN such as Amazon CloudFront sends a request to MediaTailor for HLS or DASH content. The request contains parameters from the player with information about the viewer, which is used for ad personalization.
2. MediaTailor sends a request to the ADS that contains the viewer information. The ADS chooses ads based on the viewer information and current ad campaigns. It returns the URLs to the ad creatives in a VAST or VMAP response to MediaTailor.

   If you pre-conditioned the ads, the URLs point to the pre-transcoded ads. For information about ad stitching with pre-transcoded ads, see Preconditioned ads.

3. MediaTailor manipulates the manifest to include the ad URLs returned from the ADS, transcoded to match the encoding characteristics of the origin content. If you're using preconditioned ads, it's your responsibility to ensure that the ad matches the template manifest.

   If an ad hasn't yet been transcoded to match the content, MediaTailor will skip inserting it and use MediaConvert to prepare the ad so that it's ready for the next request.
4. MediaTailor returns the fully personalized manifest to the requesting CDN or player.

The ADS tracks the ads viewed based on viewing milestones such as start of ad, middle of ad, and end of ad. As playback progresses, the player or MediaTailor sends ad tracking beacons to the ADS ad tracking URL, to record how much of an ad has been viewed. In the session initialization with MediaTailor, the player indicates whether it or MediaTailor is to send these beacons for the session.

For information about how to get started with ad insertion, see Getting started with MediaTailor.

Related services

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

Accessing MediaTailor

You can access MediaTailor using the service's console.

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

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

Pricing for MediaTailor

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

Regions for MediaTailor

To reduce data latency in your applications, MediaTailor offers regional endpoints to make your requests. To view the list of Regions in which MediaTailor is available, see Regional endpoints.

Setting up AWS Elemental MediaTailor

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

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

Topics

- Sign up for an AWS account
- Create a user with administrative access

Sign up for an AWS account

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

To sign up for an AWS account

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

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

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Getting started with MediaTailor

To get started with MediaTailor, you can choose between two tutorials—one on setting up ad insertion, and another on channel assembly. The ad insertion tutorial will guide you through sending a playback request to MediaTailor to incorporate personalized ads into your content stream. The channel assembly tutorial will show you how to view your channel's stream, including the personalized ads, directly in a browser using a playback URL.

Topics

- Getting started with MediaTailor ad insertion
- Getting started with MediaTailor channel assembly

Getting started with MediaTailor ad insertion

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

This Getting Started tutorial shows you how to perform the following tasks:

- Prepare your HLS or DASH content streams
- Configure an ad decision server (ADS) template URL
- Create a MediaTailor configuration that contains a playback endpoint
- Use your player or content delivery network (CDN) to make a playback request to MediaTailor

When you're finished, you'll be able to send a playback request to MediaTailor for personalized ad content in your stream.

Topics

- Prerequisites
- Step 1: Access AWS Elemental MediaTailor
- Step 2: Prepare a stream
- Step 3: Configure ADS request URL and query parameters
- Step 4: Create a configuration

- Step 5: Test the configuration
- Step 6: Send the playback request to AWS Elemental MediaTailor
- Step 7 (optional): Monitor AWS Elemental MediaTailor activity
- Step 8: Clean up

Prerequisites

Before you begin, be sure that you've completed the steps in Setting up AWS Elemental MediaTailor.

Step 1: Access AWS Elemental MediaTailor

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

Step 2: Prepare a stream

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

Prepare an HLS stream

HLS manifests must satisfy the following requirements:

- Manifests must be accessible on the public internet.
- Manifests must be live or video on demand (VOD).
- Manifests must have an `EXT-X-VERSION` of 3 or higher.
- For live content, manifests must contain markers to delineate ad avails. This is optional for VOD content, which can use VMAP timeoffsets instead.

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

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

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

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

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

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

  The way that you configure the ad markers in the manifest influences whether ads are inserted in a stream or replace other fragments in the stream. For more information, see the section called "Understanding ad insertion behavior".

- HLS master manifests must follow the HLS specification documented at HTTP live streaming: Master playlist tags. In particular, `#EXT-X-STREAM-INF` must include the fields `RESOLUTION`, `BANDWIDTH`, and `CODEC`.

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

Prepare a DASH stream

DASH manifests must satisfy the following requirements:

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

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

```
 <Period start="PT444806.040S" id="123586" duration="PT15.000S">
   <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
     <Event duration="1350000">
       <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="180832"
tier="4095">
```

```
          <scte35:SpliceInsert spliceEventId="4026531855"
 spliceEventCancelIndicator="false" outOfNetworkIndicator="true"
 spliceImmediateFlag="false" uniqueProgramId="1" availNum="1" availsExpected="1">
            <scte35:Program><scte35:SpliceTime ptsTime="5672624400"/></
scte35:Program>
            <scte35:BreakDuration autoReturn="true" duration="1350000"/>
          </scte35:SpliceInsert>
        </scte35:SpliceInfoSection>
      </Event>
    </EventStream>
    <AdaptationSet mimeType="video/mp4"
        ...
    </AdaptationSet>
  </Period>
```

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

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

Step 3: Configure ADS request URL and query parameters

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

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

Example Ad tag URL from an ADS

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

Where:

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

The master manifest request from the player must provide key-value pairs that correspond to the `player_params.` query parameters in the ADS request URL. For more information about configuring key-value pairs in the request to AWS Elemental MediaTailor, see Using dynamic ad variables in MediaTailor.

Enter the configured "template" URL when you create the origin server/ADS mapping in MediaTailor, in Step 4: Create a configuration.

Testing

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

Step 4: Create a configuration

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

To create a configuration (console)

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. On the **Configurations** page, choose **Create configuration**.
3. In the **Configuration** section at the bottom of the page, for **Configuration name**, enter a unique name that describes the configuration. The name is the primary identifier for the configuration. The maximum length allowed is 512 characters.
4. For **Video content source**, enter the URL prefix for the HLS master manifest or DASH manifest for this stream, minus the asset ID. For example, if the master manifest URL is `http://`

`origin-server.com/a/master.m3u8`, you would enter `http://origin-server.com/a/`. Alternatively, you can enter a shorter prefix such as `http://origin-server.com`, but then you must include the `/a/` in the asset ID in the player request for content. The maximum length is 512 characters.

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

5. For **Ad decision server**, enter the URL for your ADS. This is either the URL with variables as described in Step 3: Configure ADS request URL and query parameters, or the static VAST URL that you are using for testing purposes. The maximum length is 25,000 characters.

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

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

   For more information about the **Location** feature, see the section called “Location feature”.

7. (Optional) If your origin server produces single-period DASH manifests, choose **DASH mpd manifest origin type**, and then choose **SINGLE_PERIOD**. By default, MediaTailor handles DASH manifests as multi-period manifests. For more information, see the section called “Integrating an MPEG-DASH source”.

8. Choose **Create configuration**.

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

Step 5: Test the configuration

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

- Example: HLS

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

- Example: DASH

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

Where:

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

  Example

```
https://777788889999.mediatailor.us-east-1.amazonaws.com
```

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

  Example

```
777788889999
```

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

  Example

```
myOrigin
```

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

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

- Example: HLS

```
https://777788889999.mediatailor.us-east-1.amazonaws.com/v1/master/
AKIAIOSFODNN7EXAMPLE/myOrigin/master.m3u8
```

- Example: DASH

```
https://777788889999.mediatailor.us-east-1.amazonaws.com/v1/dash/
AKIAIOSFODNN7EXAMPLE/myOrigin/manifest.mpd
```

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

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

- Test the stream in your own player environment.

Step 6: Send the playback request to AWS Elemental MediaTailor

Configure the downstream player or CDN to send playback requests to the configuration's playback endpoint provided from AWS Elemental MediaTailor. Any player-defined dynamic variables that you used in the ADS request URL in Step 3: Configure ADS request URL and query parameters must be defined in the manifest request from the player.

Example

Assume your template ADS URL is the following.

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

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

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

```
https://777788889999.mediatailor.us-east-1.amazonaws.com/v1/master/
AKIAIOSFODNN7EXAMPLE/myOrigin/master.m3u8?ads.cust_params=viewerinfo
```

```
https://777788889999.mediatailor.us-east-1.amazonaws.com/v1/dash/AKIAIOSFODNN7EXAMPLE/
myOrigin/manifest.mpd?ads.cust_params=viewerinfo
```

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

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

For more information about configuring key-value pairs to pass to the ADS, see Using dynamic ad variables in MediaTailor.

Step 7 (optional): Monitor AWS Elemental MediaTailor activity

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

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

To allow AWS Elemental MediaTailor access to CloudWatch (console)

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

10. Edit the trust relationship to update the principal:

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

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

    The entire policy should read as follows.

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

    4. Choose **Update Trust Policy**.

Step 8: Clean up

To avoid extraneous charges, delete all unnecessary configurations.

To delete a configuration (console)

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

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

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

   AWS Elemental MediaTailor removes the configuration.

Getting started with MediaTailor channel assembly

This Getting Started tutorial shows you how to perform the following tasks:

- Create a source location, and add source content to it
- Create a channel
- Create a program list to play your channel's content on a schedule
- Add personalized ads to the channel stream using AWS Elemental MediaTailor ad insertion

When you're finished, you'll be able to open a browser, enter the playback URL for your channel, and view your channel's stream containing personalized ads.

This tutorial walks you through the basic steps to get started with MediaTailor channel assembly. For more advanced information, see Using AWS Elemental MediaTailor to create linear assembled streams.

Estimated cost

- The fee for an active channel is $0.10 per hour. You aren't charged for channels that are inactive.

Topics

- Prerequisites
- Step 1: Create a source location
- Step 2: Add VOD sources to your source location
- Step 3: Create a channel
- Step 4: Add programs to your channel's schedule
- Step 5 (optional): Use MediaTailor to insert personalized ads into your stream
- Step 6: Start your channel

- Step 7: Test your channel
- Step 8: Clean up

Prerequisites

Before you begin this tutorial, you must complete these requirements:

- Be sure that you've completed the steps in Setting up AWS Elemental MediaTailor.
- You must have assets available for both VOD source content and ad slate. You must know the path to the manifests for the assets.

> **Note**
>
> If you are using automated adaptive bitrate (ABR) or per title encoding, you must encode your assets so that all variants are the same length and have the same number of child tracks. We recommend that you use an encoding template with a minimum segment length of one second.

Step 1: Create a source location

A source location represents the origin server where your content is stored. It can be Amazon S3, a standard web server, a content delivery network (CDN), or a packaging origin, such as AWS Elemental MediaPackage.

MediaTailor fetches the content manifests from your source location, and uses them to assemble a live sliding manifest window that references the underlying content segments.

To create a source location, perform the following procedure.

To create a source location

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Source locations**.
3. On the navigation bar, choose **Create source location**.
4. Under **Source location configuration**, enter an identifier and the location of your source content:

- **Name**: An identifier for your source location, such as **my-origin**.
- **Base URL**: The base URL of the origin server where your content is hosted, such as **https://111111111111.cloudfront.net**. The URL must be in a standard HTTP URL format, prefixed with **http://** or **https://**.

5. Choose **Create source location**.

Step 2: Add VOD sources to your source location

Now that you've defined one or more source locations for your channel, you can add one or more *VOD sources*. Each VOD source represents a single piece of content, such as a single movie, an episode of a TV show, or a highlight clip.

You must create at least one *package configuration* for your VOD source. Each package configuration contains the packaged format and manifest settings for your VOD sources. You then add your package configurations to your channel to create outputs.

You can use multiple package configurations to create different channel outputs. For example, if your VOD source is packaged as both HLS and DASH, you can create two package configurations for each format. You can then use the package configuration's source groups to create two channel outputs: one for HLS, one for DASH.

To add VOD sources and create package configurations

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Source locations**.
3. In the **Source locations** pane, choose the source location that you created in the To create a source location procedure.
4. Choose **Add VOD source**.
5. Under **VOD source details**, enter a **Name** for your VOD source, such as **my-example-video**.
6. Under **Package configurations** > *`source-group-name`* enter information about the package configuration:

> **Note**
>
> Your source's package configurations must all have the same duration, as determined by the source's manifest. And, all of the sources within a package configuration must

have the same number of child streams. To meet these requirements, we recommend that you use an encoding template for your assets. We recommend that you use an encoding template with a minimum segment length of one second. MediaTailor doesn't support per title or automated adaptive bitrate streaming (ABR) because these encoding methods violate these requirements.

- **Source group**: Enter a source group name that describes this package configuration, such as HLS-4k. Make a note of this name; you'll reference it when you create your channel's output. For more information, see Using source groups with your channel's outputs.
- **Type**: Select the packaged format for this configuration. MediaTailor supports HLS and DASH.
- **Relative path**: The relative path from the source location's **Base HTTP URL** to the manifest. For example, **/my/path/index.m3u8**.

7. Choose **Add source**.
8. Repeat steps 4-7 in this procedure to add the VOD source for your ad slate.

Step 3: Create a channel

A channel assembles your sources into a live linear stream. Each channel contains one or more outputs that correspond to your VOD source's package configurations.

First you create a channel, then you add your VOD sources to the channel's schedule by creating programs.

To create a channel

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.
3. On the navigation bar, choose **Create channel**.
4. Under **Channel details**, enter details about your channel:

   - **Name**: Enter a name for your channel.
   - **Playback mode**: Determines what kind of program transitions are allowed and what happens to a program after it finishes. Use the default loop mode.

5. Choose **Next**.

6. Under **Output details**, define the settings for this output:

   - **Manifest name**: Enter a manifest name, such as ***index***. MediaTailor will append the format extension, such as .m3u8 for HLS.

     > **Note**
     >
     > You must enter a unique manifest name per channel output.

   - **Format type**: Select the streaming format for the channel. DASH and HLS are supported. Choose the format that corresponds to the package configuration that you created in Step 1: Create a source location.
   - **Source group**: Enter the name of the source group that you created in Step 1: Create a source location.

7. Under **Manifest settings**, enter additional information about your manifest settings:

   - **Manifest window (sec)**: The time window (in seconds) contained in each manifest. The minimum value is 30 seconds, and the maximum value is 3600 seconds.

8. Choose **Next**.
9. Under **Channel policy**, select **Do not attach channel policy**. This option restricts playback to only those who have access to your AWS account credentials.
10. Choose **Next**.
11. Review your settings on the **Review and create** pane.
12. Choose **Create channel**.

> **Note**
>
> Channels are created in a stopped state. Your channel won't be active until you start it.

Step 4: Add programs to your channel's schedule

Now that you have a channel, you'll add *programs* to the channel's schedule. Each program contains a VOD source from a source location in your account. The channel schedule determines the order that your programs will play in the channel's stream.

Each program can have one or more ad breaks. You insert an ad break, by specifying a VOD source to use as an ad slate. The duration of the ad break is determined by the duration of the slate. You can optionally use a server-side ad insertion server, such as MediaTailor ad insertion, to personalize your ad breaks.

To add programs to your channel's schedule

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.
3. In the **Channels** pane, choose the channel that you created in the Step 3: Create a channel procedure.
4. Under **Program details**, enter details about your program:
   - **Name**: This is the name of the program to add to your channel's schedule.
   - **Source location name**: Choose **Select an existing source location**, and select the source location that you created in the Step 1: Create a source location from the **Select a source location** drop-down menu.
   - **VOD source name**: Choose **Select an existing VOD source** and select the VOD source that you created earlier in this tutorial.
5. Under **Playback configuration**, define how and when a program is inserted in a channel's schedule:
   - **Transition type**: This value is fixed at **Relative**. The relative transition type indicates that this program occurs relative to other programs within the program list.
   - **Relative position**: If this is the first program in your channel's schedule, you can skip this setting. If it's not the first program in your channel's schedule, then choose where in the program list to append the program. You can select **Before program** or **After program**.
   - **Relative program**: If this is the first program in your schedule, you can skip this setting. If it's not the first program in your channel's schedule, the choose **Use existing program**, select the program name that you created in To add programs to your channel's schedule.
6. Select **Add ad break**. Under **Ad breaks**, configure the settings for the ad break:
   - **Slate source location name**: Choose **Select an existing source location** and choose the source location where your slate is stored that you created earlier in this tutorial.

- **VOD source name**: Choose **Select an existing VOD source** and choose the VOD source you're using for slate that you added earlier in this tutorial. The duration of the slate determines the duration of the ad break.
- For **Offset in milliseconds**: This value determines the ad break start time in milliseconds, as an offset relative to the beginning of the program. Enter any value that's less than the duration of the VOD source, and that aligns with a segment boundary on all tracks within the program's VOD source (all audio, video and closed caption tracks), otherwise the ad break will be skipped. For example, if you enter **0**, this creates a pre-roll ad break that plays before the program begins. Note: .

7. Choose **Add program**.

   For more information about programs, see Configuring ad breaks for your program.

   For more advanced information about using ads with your linear stream, see Optional configuration settings.

Step 5 (*optional*): Use MediaTailor to insert personalized ads into your stream

You now have a channel with programs. If you want, you can use MediaTailor to insert personalized ads into the ad breaks in your programs in the channel's stream.

Prerequisites

Before you proceed, you must meet the following requirements:

- You must have an ad decision server (ADS).
- You must have configured **Ad break** settings in the Adding a program to a channel's schedule procedure.

To add personalized ads to your channel's stream using MediaTailor

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Configurations**.
3. Under **Required settings**, enter the basic required information about your configuration:

   - **Name**: The name of your configuration.

- **Content source**: Enter the playback URL from your channel's output, minus the file name and extension. For advanced information about MediaTailor configuration, see Required settings.
- **Add decision server**: Enter the URL for your ADS.

4. You can optionally configure the **Configuration aliases**, **Personalization details**, and **Advanced settings**. For information about those settings, see Optional configuration settings.
5. On the navigation bar, choose **Create configuration**.

For more advanced information about using MediaTailor ad insertion, see Using AWS Elemental MediaTailor to insert ads.

Step 6: Start your channel

You now have a channel. But before you can access the channel's stream, you need to start your channel. If you try to access a channel before it is active, MediaTailor returns an HTTP 4xx error code.

Start your channel

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.
3. On the navigation bar, choose **Start**.

Step 7: Test your channel

To verify that your channel is working correctly, open a web browser and enter the URL from your channel's output. You should see your channel's stream.

In some cases, you might need to clear the cache to see the expected behavior.

Step 8: Clean up

After you've finished with the channel that you created for this tutorial, you should clean up by deleting the channel.

You'll stop incurring charges for that channel as soon as the channel status changes to stopped. To keep your channel for later, but not incur charges, you can stop the channel now and then start it again later.

To delete your channel

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.
3. Select the channel that you want to delete.
4. If your channel is running, from the **Actions** drop-down menu, choose **Stop**. You must stop your channel before you can delete it.
5. When your channel is stopped, from the **Actions** drop-down menu, choose **Delete**.

Using AWS Elemental MediaTailor to insert ads

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

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

Topics

- Supported audio and video codecs
- Understanding AWS Elemental MediaTailor ad insertion behavior
- Understanding AWS Elemental MediaTailor server-guided ad insertion
- Requirements for ad server integrations with AWS Elemental MediaTailor
- Working with AWS Elemental MediaTailor playback configurations
- Integrating a content source for MediaTailor ad insertion
- Integrating AWS Elemental MediaTailor with Google Ad Manager
- Using a CDN to optimize ad personalization and content delivery
- Customizing ad break behavior with ad suppression
- Inserting bumpers
- Inserting pre-roll ads
- Inserting slate
- Prefetching ads
- Using preconditioned ads with AWS Elemental MediaTailor
- Using dynamic ad variables in MediaTailor
- Passing AWS Elemental MediaTailor session initialization parameters to the manifest
- Reporting ad tracking data
- Overlay ads
- Ad ID decoration

Supported audio and video codecs

MediaTailor supports the following codecs.

- Audio codecs: mp4a, ac-3, and ec-3
- Video codecs: h.264 (AVC), h.265 (HEVC), av01 (AV1)

Understanding AWS Elemental MediaTailor ad insertion behavior

AWS Elemental MediaTailor stitches ads into live or video on demand (VOD) content by either replacing or inserting ads into the origin manifest. Whether ads are inserted or replaced depends on how the ad breaks are configured in the origin manifest, and whether the content is VOD or live.

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

For information about how MediaTailor stitches ads into live and VOD content, select the applicable topic.

Topics

- Ad stitching behavior for VOD
- Live ad stitching behavior

Ad stitching behavior for VOD

MediaTailor inserts or replaces ads in VOD content based on how ad markers are configured in the origin manifest, and if the ad decision server (ADS) sends VMAP responses.

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

If ad markers are present

AWS Elemental MediaTailor inserts ads where SCTE-35 ad markers are present in the origin manifest. Ad markers with an `EXT-X-CUE-OUT` value of `0` duration indicate ad insertion.

HLS ad marker guidelines

Follow these guidelines for post-roll and ad pod SCTE signaling:

Pre-roll ads

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

For example, consider the following declaration.

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

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

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

Example 2: Ad pods

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

For example, the following declaration is a valid use of `CUE-OUT/IN` to portray an ad pod.

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

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

The preceding declaration results in an output like the following.

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

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
```

If no ad markers are present

Ad markers are the recommended way of signaling ad breaks in a manifest. However, ad markers aren't required. If the manifest doesn't contain ad markers for DASH or HLS, MediaTailor makes a single call to the ADS and creates ad breaks based on the response:

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

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

> **Tip**
>
> If you want to create mid-roll ad breaks but your ADS doesn't support VMAP, then ensure that there are ad markers in the manifest. MediaTailor inserts ads at the markers, as described in the following sections.

Live ad stitching behavior

In live streams, AWS Elemental MediaTailor always performs ad replacement, preserving the total time between the ad markers as closely as possible. When ad markers include the `DURATION` attribute, MediaTailor uses the value to determine the duration of the ad break. Every `CUE-OUT` indicator must have a duration or a matching `CUE-IN` indicator in live workflows.

MediaTailor performs ad replacement for HLS and DASH live content. For information on how MediaTailor calculates ad break placement and timing, see the section called “Ad markers” and the section called “Ad markers”.

Ad selection and replacement

AWS Elemental MediaTailor includes ads from the ad decision server (ADS) VAST response as follows:

- If a duration is specified, MediaTailor selects a set of ads that fit into the duration and includes them.
- If no duration is specified, MediaTailor plays as many ads as it can until it encounters an ad marker that indicates a return to the main content.

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

- MediaTailor tries to play complete ads, without clipping or truncation.

- Whenever MediaTailor encounters an ad marker that indicates an end to the ad break, it returns to the underlying content. This can mean shortening an ad that is currently playing.
- At the end of the duration, MediaTailor returns to the underlying content.
- If MediaTailor runs out of ads to play for the duration of an ad break it either plays the slate, if one is configured, or resumes playback of the underlying content stream. This usually happens when there aren't enough transcoded ads to fill up the duration of the ad break.

> **Tip**
>
> You can define the limit of unfilled ad time allowed in an break with the personalization threshold configuration setting. For more information, see the PlaybackConfiguration reference.

Examples

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

Understanding AWS Elemental MediaTailor server-guided ad insertion

Server-guided ad insertion (HLS interstitials) is an alternative to server-side ad insertion. Rather than stitching ads directly into media playlists, ads are referenced as a separate primary playlist. This allows for faster video start times and reduced manifest latencies.

For information about how to use server-guided ad insertion with MediaTailor, select the applicable topic.

Topics

- Enable in the playback configuration
- Create a server-guided session

Enable in the playback configuration

In order to allow players to use server-guided ad insertion, you must set `Insertion Mode` to `PLAYER_SELECT` in the MediaTailor playback configuration. This allows players to select either stitched or guided ad insertion at session-initialization time.

Create a server-guided session

When creating playback sessions, choose guided mode. The way to do this depends on whether your players use implicit or explicit sessions.

Implicitly created server-guided sessions

Append `aws.insertionMode=GUIDED` to the HLS parent manifest request. Example:

```
playback-endpoint/v1/master/hashed-account-id/origin-id/index.m3u8?
aws.insertionMode=GUIDED
```

Where:

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

  Example

  ```
  https://777788889999.mediatailor.us-east-1.amazonaws.com
  ```

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

  Example

  ```

  ```

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

  Example

  ```
  myOrigin
  ```

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

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

- Example:

```
https://777788889999.mediatailor.us-east-1.amazonaws.com/v1/master/
AKIAIOSFODNN7EXAMPLE/myOrigin/index.m3u8?aws.insertionMode=GUIDED
```

Explicitly created server-guided sessions

Add `insertionMode=GUIDED` to JSON metadata the player sends in the HTTP POST to the MediaTailor configuration's session-initialization prefix endpoint.

The following example shows the structure of the JSON metadata:

```
{
  # other keys, e.g. "adsParams"
  "insertionMode": "GUIDED"       # this can be either GUIDED or STITCHED
}
```

With this initialization metadata, the playback session will use serer-guided ad insertion.

Requirements for ad server integrations with AWS Elemental MediaTailor

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

AWS Elemental MediaTailor supports VAST and VMAP responses from ad decision servers. AWS Elemental MediaTailor also supports the proxying of VPAID metadata through our client-side reporting API for client-side ad insertion. For information about client-side reporting, see Client-side ad tracking.

MediaTailor supports the following versions of VAST, VMAP, and VPAID:

- VAST 2.0 and 3.0
- VMAP 1.0
- VPAID 2.0

VAST requirements

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

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

Targeting

To target specific players for your ads, you can create templates for your ad tags and URLs. For more information, see Using dynamic ad variables in MediaTailor.

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 Using session variables.

Ad calls

AWS Elemental MediaTailor calls your VAST ads URL as defined in your configuration. It substitutes any player-specific or session-specific parameters when making the ad call. MediaTailor follows up to seven levels of VAST wrappers and redirects in the VAST response. In live streaming scenarios, MediaTailor makes ad calls simultaneously at the ad avail start for connected players. In practice, due to jitter, these ad calls can be spread out over a few seconds. Make sure that your ad server can handle the number of concurrent connections this type of calling requires. MediaTailor supports prefetching VAST responses for live workflows. For more information, see Prefetching ads.

Creative handling

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

For a list of supported formats for media file inputs, see the **MP4** row of Supported input formats in the *AWS Elemental MediaConvert User Guide*.

Creative indexing

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

The following example declaration shows the creative ID.

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

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

VAST extensions provided by ad-serving partners

To help prevent collisions with creative IDs, you can use extensions provided by ad-serving partners for the VAST response. MediaTailor supports extensions from SpringServe, Publica, and FreeWheel. When you enable VAST extension overrides, MediaTailor replaces the default creative ID with the extension value.

To enable this feature, submit an AWS Support ticket to request VAST extension-based creative IDs to be enabled. Include the following information in the Support ticket:

- AWS Region

- AWS account ID
- MediaTailor playback configuration names

To validate that VAST extension-based creative IDs are enabled on your account, we recommend that you also request `RAW_ADS_RESPONSE` logging to be enabled on a staging or testing playback configuration. With logging, you can view the original VAST response that the ADS receives and confirm that the correct creative IDs are used.

VPAID requirements

VPAID allows publishers to serve highly interactive video ads and to provide viewability metrics on their monetized streams. For information about VPAID, see the VPAID specification.

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

To use VPAID, follow these guidelines:

- Configure an MP4 slate for your VPAID creatives. AWS Elemental MediaTailor fills the VPAID ad slots with your configured slate, and provides VPAID ad metadata for the client player to use to run the interactive ads. If you don't have a slate configured, when a VPAID ad appears, MediaTailor provides the ad metadata through client-side reporting as usual. It also logs an error in CloudWatch about the missing slate. For more information, see Inserting slate and Creating a configuration.
- Use client-side reporting. AWS Elemental MediaTailor supports VPAID through our client-side reporting API. For more information, see Client-side ad tracking.

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

Working with AWS Elemental MediaTailor playback configurations

This section covers the key tasks for managing MediaTailor playback configurations. You can learn how to create a new configuration to set up content streams and provide access for playback devices, view details of an existing configuration, edit a configuration to update settings like origin servers and ad decision servers, and delete a configuration that is no longer needed.

Topics

- Creating a configuration
- Viewing a configuration
- Editing a configuration
- Deleting a configuration

Creating a configuration

This topic shows how to create a configuration to start receiving content streams. It also shows how to provide an access point for downstream playback devices to request content.

You can use the AWS Elemental MediaTailor console, the AWS Command Line Interface (AWS CLI)>, or the MediaTailor API to create a configuration. For information about creating a configuration through the AWS CLI or MediaTailor API, see the *AWS Elemental MediaTailor API Reference*..

When you create a configuration, don't put sensitive identifying information into free-form fields such as the **Configuration name** field. Identifying information can include things like customer account numbers. In addition, don't use identifying information when you work in the MediaTailor console, REST API, the AWS CLI, or AWS SDKs. Any data that you enter into MediaTailor might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

To add a configuration (console)

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. On the **Configurations** page, choose **Create configuration**.
3. Complete the configuration and additional configuration fields as described in the following topics:

- Required settings
- Optional configuration settings

4. Choose **Create configuration**.

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

5. (Recommended) Set up a CDN with AWS Elemental MediaTailor for manifest and reporting requests. You can use the configuration playback URLs for the CDN setup. For information about setting up a CDN for manifest and reporting requests, see Integrating a CDN.

Required settings

When you create a configuration, you must include the following required settings.

Name

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

Content source

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

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

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

Ad decision server

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

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

Optional configuration settings

You can optionally configure **configuration aliases**, **personalization details**, and **advanced settings** in the MediaTailor console, MediaTailor API, or the AWS Command Line Interface (AWS CLI).

Configuration aliases

The following are optional configuration aliases that you can configure in the MediaTailor console, or with the MediaTailor API.

Player parameter variable

For dynamic domain configuration during session initialization, add one or more player parameter variables.

For more information about using player parameter variables to dynamically configure domains, see Using domain variables to configure multiple content and ad sources.

Log configuration

The following are log configuration settings.

Percent enabled

Sets the percentage of playback configuration session logs that MediaTailor writes to CloudWatch Logs. For example, if your playback configuration has 1000 sessions, and you set percent enabled to **60**, MediaTailor writes 600 session logs to CloudWatch Logs.

When you enable this option, MediaTailor automatically creates a service-linked role that allows MediaTailor to write and manage session logs in your CloudWatch Logs account. For more information, see Using service-linked roles for MediaTailor.

Logging strategies

Indicates the method used for collecting logs that MediaTailor emits. To send logs directly to CloudWatch Logs, choose `LEGACY_CLOUDWATCH`. To send logs to CloudWatch Logs, which then vends logs to your destination of choice, choose `VENDED_LOGS`. Supported destinations are a CloudWatch Logs log group, Amazon S3 bucket, and Amazon Data Firehose stream.

Additional set up is required for vended logs. For set up, see Using vended logs.

Ad conditioning

The following determine what actions MediaTailor takes to condition ads before stitching them into a content stream.

Streaming media file conditioning

Determines the logic that MediaTailor uses when deciding which ads to stitch.

- When **Streaming media file conditioning** is set to **Transcode**, MediaTailor transcodes the media files with `progressive` delivery and stitches them into the manifest. If there aren't enough ads with `progressive` delivery media files to fill the avail, MediaTailor transcodes and uses those with `streaming` delivery.
- When **Streaming media file conditioning** is set to **None**, MediaTailor stitches ads with `streaming` delivery media files into the manifest without transcoding them. If there aren't enough ads with `streaming` delivery media files to fill the avail, MediaTailor transcodes and uses those with `progressive` delivery.

Personalization details

The following are personalization details that you can configure in the MediaTailor console or with the MediaTailor API.

Slate ad

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

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

Start bumper

The URL of the start bumper asset location. Bumpers are short video or audio clips that play at the beginning or end of an ad break. They can be stored on Amazon's S3, or a different storage service. To learn more about bumpers, see Inserting bumpers.

End bumper

The URL of the end bumper asset location. Bumpers are short video or audio clips that play at the beginning or end of an ad break. They can be stored on Amazon's S3, or a different storage service. To learn more about bumpers, see Inserting bumpers.

Personalization threshold

Defines the maximum duration of underfilled ad time (in seconds) allowed in an ad break. If the duration of underfilled ad time exceeds the personalization threshold, then the personalization of the ad break is abandoned and the underlying content is shown. For example, if the personalization threshold is 3 seconds and there would be 4 seconds of slate in an ad break, then personalization of the ad break is abandoned and the underlying content is shown. This feature applies to *ad replacement* in live and VOD streams, rather than ad insertion, because it relies on an underlying content stream. For more information about ad break behavior, including ad replacement and insertion, see Understanding AWS Elemental MediaTailor ad insertion behavior.

Live pre-roll ad decision server

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

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

For information about how pre-roll works, see Inserting pre-roll ads.

Live pre-roll maximum allowed duration

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

Avail suppression mode

Sets the mode for avail suppression, also known as ad suppression. By default, ad suppression is off and MediaTailor fills all with ads or slate. When the mode is set to `BEHIND_LIVE_EDGE`, ad suppression is active and MediaTailor doesn't fill ad breaks on or behind the avail suppression value time in the manifest lookback window. When the mode is set to `AFTER_LIVE_EDGE`, ad suppression is active. MediaTailor doesn't fill ad breaks on or behind the avail suppression period, which is the live edge plus the avail suppression value plus buffer time.

Avail suppression value

The avail suppression value is a live edge offset time in `HH:MM:SS`. MediaTailor won't fill ad breaks on or behind this time in the manifest lookback window.

Insertion Mode

Insertion Mode controls whether players can use stitched or guided ad insertion. The default, `STITCHED_ONLY`, forces all player sessions to use stitched (server-side) ad insertion. Setting

InsertionMode to PLAYER_SELECT allows players to select either stitched or guided ad insertion at session-initialization time. The default for players that do not specify an insertion mode is stitched.

Advanced settings

The following are optional settings are advanced. You can configure these in the MediaTailor console, with the AWS Command Line Interface (AWS CLI), or using the MediaTailor API.

CDN content segment prefix

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

For more information about integrating MediaTailor with a CDN, see Using a CDN to optimize ad personalization and content delivery.

CDN ad segment prefix

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

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

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

For more information about integrating MediaTailor with a CDN, see Using a CDN to optimize ad personalization and content delivery.

DASH origin manifest type

If your origin server produces single-period DASH manifests, open the dropdown list and choose **SINGLE_PERIOD**. By default, MediaTailor handles DASH manifests as multi-period manifests. For more information, see the section called “Integrating an MPEG-DASH source”.

DASH mpd location

(Optional as needed for DASH) The media presentation description (mpd) location. Choose **DISABLED** for the following situation:

- You set up CDN routing rules for accessing MediaTailor manifests.
- You use client-side reporting, or your player supports sticky HTTP redirects.

For more information about the **Location** feature, see the section called “Location feature”.

Transcode profile name

The name that associates this configuration with a custom transcode profile. This name overrides the dynamic transcoding defaults of MediaTailor. Complete this field only if you have already set-up custom profiles with the help of AWS Support.

Ad marker passthrough

For HLS, enables or disables ad marker passthrough. When ad marker passthrough is enabled, MediaTailor passes through `EXT-X-CUE-IN`, `EXT-X-CUE-OUT`, and `EXT-X-SPLICEPOINT-SCTE35` ad markers from the origin manifest to the MediaTailor personalized manifest. No logic is applied to the ad marker values; they are passed from the origin manifest to the personalized manifest as-is. For example, if `EXT-X-CUE-OUT` has a value of `60` in the origin manifest, but no ads are placed, MediaTailor won't change the value to `0` in the personalized manifest.

Viewing a configuration

In addition to the values provided when the configuration was created, MediaTailor displays the name of the configuration, playback endpoints, and relevant access URLs. To view a configuration, use the following procedure.

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.

Editing a configuration

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

To edit a configuration

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

Deleting a configuration

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

To delete a configuration

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

Integrating a content source for MediaTailor ad insertion

This topic covers integrating different types of video content sources with MediaTailor. MediaTailor supports both HLS and DASH streaming protocols for live and on-demand content. The service can perform ad insertion or replacement during designated ad breaks, and has specific requirements for the structure and formatting of the input video manifests to enable these capabilities. The following topics provide details on the input source requirements and steps for integrating HLS and DASH content with MediaTailor to enable personalized ad experiences.

Topics

- Input source requirements for MediaTailor ad insertion
- Integrating an HLS source

- Integrating an MPEG-DASH source
- Securing AWS Elemental MediaTailor origin interactions with SigV4

Input source requirements for MediaTailor ad insertion

A input source must meet the following requirements to work with MediaTailor:

- Use Apple HLS (HTTP Live Streaming) or MPEG DASH (Dynamic Adaptive Streaming over HTTP)
- Use live streaming or video on demand (VOD)
- Be accessible on the public internet and have a public IP address
- Contain ad markers in one of the formats described in the Getting started with MediaTailor ad insertion tutorial

Integrating an HLS source

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

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

Topics

- HLS supported ad markers
- Enabling ad marker passthrough
- HLS manifest tag handling
- HLS manifest examples

HLS supported ad markers

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

EXT-X-ASSET

The EXT-X-ASSET tag contains metadata that's used by the ad decision server (ADS) to personalize content for the viewer. EXT-X-ASSET parameters are comma-separated key-value pairs.

To use this tag, you must meet the following requirements:

- You must URL-encode the EXT-X-ASSET *values* in the origin manifest. The following example shows the EXT-X-ASSET tag with keys and URL-encoded values.

```
#EXT-X-ASSET:GENRE=CV,CAID=12345678,EPISODE="Episode%20Name
%20Date",SEASON="Season%20Name%20and%20Number",SERIES="Series%2520Name"
```

- You must include the dynamic [asset.] variable and the *keys* in your MediaTailor ADS configuration. The following example shows an MediaTailor ADS configuration using the dynamic [asset.] variable and keys.

```
https://myads.com/stub?
c=[asset.GENRE]&g=[asset.CAID]&e=[asset.EPISODE]&s=[asset.SEASON]&k=[asset.SERIES]
```

Example VAST request

The following example shows a VAST GET request to an ADS.

```
https://myads.com/stub?c=CV&g=12345678&e=Episode%20Name%20Date&s=Season
%20Name%20and%20Number&k=Series%2520Name
```

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

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

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

```
    #EXT-X-CUE-IN
```

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

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

EXT-X-DATERANGE

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

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

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

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

  Example

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

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

  Example

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

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

  Example

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

EXT-X-SPLICEPOINT-SCTE35

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

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

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

Enabling ad marker passthrough

By default for HLS, MediaTailor personalized manifests don't include the SCTE-35 ad markers from the origin manifests. When ad marker passthrough is enabled, MediaTailor passes through the following ad markers from origin manifests into personalized manifests:

- EXT-X-CUE-IN
- EXT-X-CUE-OUT

- EXT-X-SPLICEPOINT-SCTE35

Ad marker passthrough is an optional setting. Use ad marker passthrough if you want the SCTE ad markers to be included in the MediaTailor personalized manifest. Common use cases include the following:

- Content replacement - Perform content replacement or content restriction.
- Ad tracking - Trigger ad tracking information based on the presence or absence of one or more ad markers.
- Player settings - Enable scrubbing or countdown timer functionality in the player's UI, based on the presence or absence of ad markers.

> **Note**
>
> MediaTailor doesn't change the values for these markers. For example, if `EXT-X-CUE-OUT` has a value of `60` in the origin manifest, but no ads are placed, MediaTailor won't change the value to `0` in the personalized manifest.

Enable ad marker passthrough

You can enable ad marker passthrough using the AWS Management Console or the AWS Command Line Interface (AWS CLI).

To enable ad marker passthrough using the console

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. Select either **New Configuration** or **Edit Configuration**.
3. In the **Advanced Settings** section, select **Enable** from the drop down menu.

To enable ad marker passthrough using the AWS Command Line Interface (AWS CLI)

Use the put-playback-configuration command.

HLS manifest tag handling

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

EXT-X-CUE tags

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

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

EXT-X-DATERANGE tags

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

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

EXT-X-KEY tags

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

Unrecognized tags

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

HLS manifest examples

The following sections provide examples of HLS origin manifests and personalized manifests.

HLS origin manifest examples

The following example shows an HLS master manifest that AWS Elemental MediaTailor received by HLS from the content origin.

```
#EXTM3U
    #EXT-X-VERSION:3
    #EXT-X-INDEPENDENT-SEGMENTS
    #EXT-X-STREAM-INF:BANDWIDTH=2665726,AVERAGE-
BANDWIDTH=2526299,RESOLUTION=960x540,FRAME-
RATE=29.970,CODECS="avc1.640029,mp4a.40.2",SUBTITLES="subtitles"
    index_1.m3u8
    #EXT-X-STREAM-INF:BANDWIDTH=3956044,AVERAGE-
BANDWIDTH=3736264,RESOLUTION=1280x720,FRAME-
RATE=29.970,CODECS="avc1.640029,mp4a.40.2",SUBTITLES="subtitles"
    index_2.m3u8
    #EXT-X-STREAM-INF:BANDWIDTH=995315,AVERAGE-
BANDWIDTH=951107,RESOLUTION=640x360,FRAME-
RATE=29.970,CODECS="avc1.4D401E,mp4a.40.2",SUBTITLES="subtitles"
    index_3.m3u8
    #EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-
ID="subtitles",NAME="caption_1",DEFAULT=YES,AUTOSELECT=YES,FORCED=NO,LANGUAGE="eng",URI="index_
```

The following example shows an HLS media manifest that AWS Elemental MediaTailor received by HLS from the content origin. This example uses `EXT-X-CUE-OUT` and `EXT-X-CUE-IN` tags to describe ad avail opportunities.

```
#EXTM3U
    #EXT-X-VERSION:3
    #EXT-X-TARGETDURATION:7
    #EXT-X-MEDIA-SEQUENCE:8779957
    #EXTINF:6.006,
    index_1_8779957.ts?m=1566416212
    #EXTINF:6.006,
    index_1_8779958.ts?m=1566416212
    #EXTINF:5.372,
    index_1_8779959.ts?m=1566416212
    #EXT-OATCLS-SCTE35:/DAlAAAAAsvhAP/wFAXwAAAGf+/+AdLfiP4AG3dAAAEBAQAAXytxmQ==
    #EXT-X-CUE-OUT:20.020
```

```
    #EXTINF:0.634,
    index_1_8779960.ts?m=1566416212
    #EXT-X-CUE-OUT-CONT:ElapsedTime=0.634,Duration=21,SCTE35=/DAlAAAAAsvhAP/wFAXwAAAGf
+/+AdLfiP4AG3dAAAEBAQAAXytxmQ==
    #EXTINF:6.006,
    index_1_8779961.ts?m=1566416212
    #EXT-X-CUE-OUT-CONT:ElapsedTime=6.640,Duration=21,SCTE35=/DAlAAAAAsvhAP/wFAXwAAAGf
+/+AdLfiP4AG3dAAAEBAQAAXytxmQ==
    #EXTINF:6.006,
    index_1_8779962.ts?m=1566416212
    #EXT-X-CUE-OUT-CONT:ElapsedTime=12.646,Duration=21,SCTE35=/DAlAAAAAsvhAP/wFAXwAAAGf
+/+AdLfiP4AG3dAAAEBAQAAXytxmQ==
    #EXTINF:6.006,
    index_1_8779963.ts?m=1566416212
    #EXT-X-CUE-OUT-CONT:ElapsedTime=18.652,Duration=21,SCTE35=/DAlAAAAAsvhAP/wFAXwAAAGf
+/+AdLfiP4AG3dAAAEBAQAAXytxmQ==
    #EXTINF:1.368,
    index_1_8779964.ts?m=1566416212
    #EXT-X-CUE-IN
    #EXTINF:4.638,
    index_1_8779965.ts?m=1566416212
    #EXTINF:6.006,
    index_1_8779966.ts?m=1566416212
    #EXTINF:6.006,
    index_1_8779967.ts?m=1566416212
    #EXTINF:6.006,
    index_1_8779968.ts?m=1566416212
```

HLS personalized manifest examples

The following example shows an HLS master manifest that AWS Elemental MediaTailor personalized.

```
#EXTM3U
    #EXT-X-VERSION:3
    #EXT-X-MEDIA:LANGUAGE="eng",AUTOSELECT=YES,FORCED=NO,TYPE=SUBTITLES,URI="../../../
manifest/43f3e412052f2808dd84ea1da90e92e914edddee/external-
canary-hls/ee1696a8-4f7f-4c4c-99de-9821131847e8/3.m3u8",GROUP-
ID="subtitles",DEFAULT=YES,NAME="caption_1"
    #EXT-X-INDEPENDENT-SEGMENTS
    #EXT-X-STREAM-INF:CODECS="avc1.640029,mp4a.40.2",AVERAGE-
BANDWIDTH=2526299,RESOLUTION=960x540,SUBTITLES="subtitles",FRAME-
RATE=29.97,BANDWIDTH=2665726
```

```
    ../../../manifest/43f3e412052f2808dd84ea1da90e92e914edddee/external-canary-hls/
ee1696a8-4f7f-4c4c-99de-9821131847e8/0.m3u8
    #EXT-X-STREAM-INF:CODECS="avc1.640029,mp4a.40.2",AVERAGE-
BANDWIDTH=3736264,RESOLUTION=1280x720,SUBTITLES="subtitles",FRAME-
RATE=29.97,BANDWIDTH=3956044
    ../../../manifest/43f3e412052f2808dd84ea1da90e92e914edddee/external-canary-hls/
ee1696a8-4f7f-4c4c-99de-9821131847e8/1.m3u8
    #EXT-X-STREAM-INF:CODECS="avc1.4D401E,mp4a.40.2",AVERAGE-
BANDWIDTH=951107,RESOLUTION=640x360,SUBTITLES="subtitles",FRAME-
RATE=29.97,BANDWIDTH=995315
    ../../../manifest/43f3e412052f2808dd84ea1da90e92e914edddee/external-canary-hls/
ee1696a8-4f7f-4c4c-99de-9821131847e8/2.m3u8
```

The following example shows a media master manifest that AWS Elemental MediaTailor personalized.

```
#EXTM3U
    #EXT-X-VERSION:6
    #EXT-X-TARGETDURATION:7
    #EXT-X-MEDIA-SEQUENCE:8779957
    #EXT-X-DISCONTINUITY-SEQUENCE:0
    #EXTINF:6.006,
    https://777788889999.mediapackage.us-west-2.amazonaws.com/out/v1/
e309ffd02ba8498d864dcaacff7a5ad9/index_1_8779957.ts?m=1566416212
    #EXTINF:6.006,
    https://777788889999.mediapackage.us-west-2.amazonaws.com/out/v1/
e309ffd02ba8498d864dcaacff7a5ad9/index_1_8779958.ts?m=1566416212
    #EXTINF:5.372,
    https://777788889999.mediapackage.us-west-2.amazonaws.com/out/v1/
e309ffd02ba8498d864dcaacff7a5ad9/index_1_8779959.ts?m=1566416212
    #EXT-X-DISCONTINUITY
    #EXTINF:3.066667,
    ../../../../segment/43f3e412052f2808dd84ea1da90e92e914edddee/external-canary-hls/
ee1696a8-4f7f-4c4c-99de-9821131847e8/0/8779960
    #EXTINF:3.0,
    ../../../../segment/43f3e412052f2808dd84ea1da90e92e914edddee/external-canary-hls/
ee1696a8-4f7f-4c4c-99de-9821131847e8/0/8779961
    #EXTINF:3.0,
    ../../../../segment/43f3e412052f2808dd84ea1da90e92e914edddee/external-canary-hls/
ee1696a8-4f7f-4c4c-99de-9821131847e8/0/8779962
    #EXTINF:3.0,
```

```
    ../../../../segment/43f3e412052f2808dd84ea1da90e92e914edddee/external-canary-hls/
ee1696a8-4f7f-4c4c-99de-9821131847e8/0/8779963
    #EXTINF:2.966667,
    ../../../../segment/43f3e412052f2808dd84ea1da90e92e914edddee/external-canary-hls/
ee1696a8-4f7f-4c4c-99de-9821131847e8/0/8779964
    #EXT-X-DISCONTINUITY
    #EXTINF:6.006,
    https://777788889999.mediapackage.us-west-2.amazonaws.com/out/v1/
e309ffd02ba8498d864dcaacff7a5ad9/index_1_8779963.ts?m=1566416212
    #EXTINF:1.368,
    https://777788889999.mediapackage.us-west-2.amazonaws.com/out/v1/
e309ffd02ba8498d864dcaacff7a5ad9/index_1_8779964.ts?m=1566416212
    #EXTINF:4.638,
    https://777788889999.mediapackage.us-west-2.amazonaws.com/out/v1/
e309ffd02ba8498d864dcaacff7a5ad9/index_1_8779965.ts?m=1566416212
    #EXTINF:6.006,
    https://777788889999.mediapackage.us-west-2.amazonaws.com/out/v1/
e309ffd02ba8498d864dcaacff7a5ad9/index_1_8779966.ts?m=1566416212
    #EXTINF:6.006,
    https://777788889999.mediapackage.us-west-2.amazonaws.com/out/v1/
e309ffd02ba8498d864dcaacff7a5ad9/index_1_8779967.ts?m=1566416212
    #EXTINF:6.006,
    https://777788889999.mediapackage.us-west-2.amazonaws.com/out/v1/
e309ffd02ba8498d864dcaacff7a5ad9/index_1_8779968.ts?m=1566416212
```

Integrating an MPEG-DASH source

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

Input manifests must have the following:

- SCTE-35 event streams with splice info settings for either `splice insert` or `time signal`. The settings can be provided in clear XML or in base64-encoded binary.
- `Segment templates` with `segment timelines`.

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

- Period start times, specified in the `start` attribute.

- Values of `presentationTimeOffset` in the segment templates of the period representations.

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

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

Topics

- DASH ad markers
- DASH ad avail duration
- DASH manifest segment numbering
- Live DASH manifest examples
- VOD DASH manifest examples
- DASH location feature

DASH ad markers

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

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

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

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

Clear XML

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

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

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

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

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

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

```
  <Period start="PT346530.250S" id="178443" duration="PT61.561S">
          <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
            <Event duration="5310000">
              <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="183003"
 tier="4095">
                <scte35:TimeSignal>
                  <scte35:SpliceTime ptsTime="3442857000"/>
                </scte35:TimeSignal>
```

```
                <scte35:SegmentationDescriptor segmentationEventId="1414668"
  segmentationEventCancelIndicator="false"
  segmentationDuration="8100000" segmentationTypeId="52" segmentNum="0"
  segmentsExpected="0">
                  <scte35:DeliveryRestrictions webDeliveryAllowedFlag="false"
  noRegionalBlackoutFlag="false" archiveAllowedFlag="false" deviceRestrictions="3"/>
                  <scte35:SegmentationUpid segmentationUpidType="12"
  segmentationUpidLength="2">0100</scte35:SegmentationUpid>
                </scte35:SegmentationDescriptor>
              </scte35:SpliceInfoSection>
            </Event>
```

Base64-encoded binary

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

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

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

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

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

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

```
          "table_id": 252,
          "splice_command_type": 5
        }
      }
```

For multi-period DASH manifests, AWS Elemental MediaTailor uses the first `Event` that indicates ad placement in an event stream, and it ignores any additional `Event` markers in the stream. For single-period DASH manifests, MediaTailor considers each `Event`.

DASH ad avail duration

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

- If the ad avail specifies a duration, MediaTailor includes as many ads as it can fit inside the duration boundary, without overwriting content that follows.
- If no duration is provided, MediaTailor includes ads until it reaches the end of the ad avail. For multi-period manifests, this is the end of the period. For single-period manifests, this is the end of the event. MediaTailor doesn't play ads past the end of the ad avail and, when it encounters the end, truncates the current ad instead of overwriting the content that follows.

How AWS Elemental MediaTailor looks for the ad avail duration

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

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

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

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

```
<Period start="PT444806.040S" id="123586" duration="PT15.000S">
        <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
          <Event duration="1350000">
```

```
              <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="180832"
 tier="4095">
                <scte35:SpliceInsert spliceEventId="4026531855"
 spliceEventCancelIndicator="false" outOfNetworkIndicator="true"
 spliceImmediateFlag="false" uniqueProgramId="1" availNum="1" availsExpected="1">
                  <scte35:Program><scte35:SpliceTime ptsTime="5672624400"/></
scte35:Program>
                  <scte35:BreakDuration autoReturn="true" duration="1350000"/>
                </scte35:SpliceInsert>
              </scte35:SpliceInfoSection>
            </Event>
              ...
```

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

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

DASH manifest segment numbering

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

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

```
        <SegmentTemplate initialization="index_subtitles_4_0_init.mp4?
m=1532451703" media="index_subtitles_4_0_$Number$.mp4?m=1532451703"
 presentationTimeOffset="1062336677920" startNumber="2349899" timescale="90000">
                <SegmentTimeline>
                  <S d="540540" r="2" t="1062338840080"/>
                  <S d="69069" t="1062340461700"/>
                </SegmentTimeline>
              </SegmentTemplate>
```

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

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

Live DASH manifest examples

This section provides examples of live DASH manifests. Each example lists a manifest as received from the origin server and after MediaTailor has personalized the manifest with ads.

Topics

- DASH manifest splice insert example
- DASH manifest time signal example
- DASH manifest Base64-encoded binary example with single-period input

DASH manifest splice insert example

DASH origin manifest example for splice insert

The following example from an MPD manifest shows an ad avail in a manifest received by DASH from the content origin. T

```
        <Period start="PT173402.036S" id="46041">
```

```
          <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
            <Event duration="9450000">
              <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="183265"
 tier="4095">
                <scte35:SpliceInsert spliceEventId="99"
 spliceEventCancelIndicator="false" outOfNetworkIndicator="true"
 spliceImmediateFlag="false" uniqueProgramId="1" availNum="1" availsExpected="1">
                  <scte35:Program><scte35:SpliceTime ptsTime="7835775000"/></
scte35:Program>
                  <scte35:BreakDuration autoReturn="true" duration="9450000"/>
                </scte35:SpliceInsert>
              </scte35:SpliceInfoSection>
            </Event>
          </EventStream>
          <AdaptationSet mimeType="video/mp4" segmentAlignment="true"
 subsegmentAlignment="true" startWithSAP="1" subsegmentStartsWithSAP="1"
 bitstreamSwitching="true">
            <Representation id="1" width="640" height="360" frameRate="30/1"
 bandwidth="749952" codecs="avc1.4D4029">
              <SegmentTemplate timescale="30" media="index_video_1_0_$Number
$.mp4?m=1531257079" initialization="index_video_1_0_init.mp4?m=1531257079"
 startNumber="46042" presentationTimeOffset="5202061">
                <SegmentTimeline>
                  <S t="5202061" d="115"/>
                  <S t="5202176" d="120" r="4"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation id="2" width="1280" height="720" frameRate="30/1"
 bandwidth="2499968" codecs="avc1.4D4029">
              <SegmentTemplate timescale="30" media="index_video_3_0_$Number
$.mp4?m=1531257079" initialization="index_video_3_0_init.mp4?m=1531257079"
 startNumber="46042" presentationTimeOffset="5202061">
                <SegmentTimeline>
                  <S t="5202061" d="115"/>
                  <S t="5202176" d="120" r="4"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation id="3" width="1920" height="1080" frameRate="30/1"
 bandwidth="4499968" codecs="avc1.4D4029">
              <SegmentTemplate timescale="30" media="index_video_5_0_$Number
$.mp4?m=1531257079" initialization="index_video_5_0_init.mp4?m=1531257079"
 startNumber="46042" presentationTimeOffset="5202061">
```

```
                <SegmentTimeline>
                  <S t="5202061" d="115"/>
                  <S t="5202176" d="120" r="4"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet mimeType="audio/mp4" segmentAlignment="0" lang="eng">
            <Representation id="4" bandwidth="128858" audioSamplingRate="44100"
 codecs="mp4a.40.2">
              <SegmentTemplate timescale="44100" media="index_audio_2_0_$Number
$.mp4?m=1531257079" initialization="index_audio_2_0_init.mp4?m=1531257079"
 startNumber="46042" presentationTimeOffset="7647030507">
                <SegmentTimeline>
                  <S t="7647030507" d="168959"/>
                  <S t="7647199468" d="176127" r="1"/>
                  <S t="7647551723" d="177151"/>
                  <S t="7647728875" d="176127" r="1"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation id="5" bandwidth="128858" audioSamplingRate="44100"
 codecs="mp4a.40.2">
              <SegmentTemplate timescale="44100" media="index_audio_4_0_$Number
$.mp4?m=1531257079" initialization="index_audio_4_0_init.mp4?m=1531257079"
 startNumber="46042" presentationTimeOffset="7647030507">
                <SegmentTimeline>
                  <S t="7647030507" d="168959"/>
                  <S t="7647199468" d="176127" r="1"/>
                  <S t="7647551723" d="177151"/>
                  <S t="7647728875" d="176127" r="1"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation id="6" bandwidth="128858" audioSamplingRate="44100"
 codecs="mp4a.40.2">
              <SegmentTemplate timescale="44100" media="index_audio_6_0_$Number
$.mp4?m=1531257079" initialization="index_audio_6_0_init.mp4?m=1531257079"
 startNumber="46042" presentationTimeOffset="7647030507">
                <SegmentTimeline>
                  <S t="7647030507" d="168959"/>
                  <S t="7647199468" d="176127" r="1"/>
                  <S t="7647551723" d="177151"/>
                  <S t="7647728875" d="176127" r="1"/>
```

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

DASH personalized response example for splice insert

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

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

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

```
            <Representation bandwidth="2000000" codecs="avc1.64001f" height="540"
 id="4" width="960">
              <SegmentTemplate initialization="EXAMPLE_PRODUCT_540p_7init.mp4"
 media="EXAMPLE_PRODUCT_540p_7_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
            </Representation>
            <Representation bandwidth="1350000" codecs="avc1.64001e" height="396"
 id="5" width="704">
              <SegmentTemplate initialization="EXAMPLE_PRODUCT_396p_6init.mp4"
 media="EXAMPLE_PRODUCT_396p_6_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
            </Representation>
            <Representation bandwidth="900000" codecs="avc1.64001e" height="396" id="6"
 width="704">
              <SegmentTemplate initialization="EXAMPLE_PRODUCT_396p_5init.mp4"
 media="EXAMPLE_PRODUCT_396p_5_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
            </Representation>
            <Representation bandwidth="600000" codecs="avc1.64001e" height="396" id="7"
 width="704">
              <SegmentTemplate initialization="EXAMPLE_PRODUCT_396p_4init.mp4"
 media="EXAMPLE_PRODUCT_396p_4_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
            </Representation>
            <Representation bandwidth="450000" codecs="avc1.640016" height="288" id="8"
 width="512">
              <SegmentTemplate initialization="EXAMPLE_PRODUCT_288p_3init.mp4"
 media="EXAMPLE_PRODUCT_288p_3_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
            </Representation>
            <Representation bandwidth="300000" codecs="avc1.640016" height="288" id="9"
 width="512">
              <SegmentTemplate initialization="EXAMPLE_PRODUCT_288p_2init.mp4"
 media="EXAMPLE_PRODUCT_288p_2_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
            </Representation>
            <Representation bandwidth="200000" codecs="avc1.640016" height="288"
 id="10" width="512">
```

```
              <SegmentTemplate initialization="EXAMPLE_PRODUCT_288p_1init.mp4"
 media="EXAMPLE_PRODUCT_288p_1_$Number%09d$.mp4" startNumber="1"
 timescale="90000"><SegmentTimeline><S d="180000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a1_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a1_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"/>
            <Representation audioSamplingRate="48000" bandwidth="128000"
 codecs="mp4a.40.2" id="11"><SegmentTemplate
 initialization="EXAMPLE_PRODUCT_audio_aac_a1_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a1_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"><SegmentTimeline><S d="96000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate></Representation>
          </AdaptationSet>
          <AdaptationSet lang="enm" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a2_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a2_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"/>
            <Representation audioSamplingRate="48000" bandwidth="128000"
 codecs="mp4a.40.2" id="12"><SegmentTemplate
 initialization="EXAMPLE_PRODUCT_audio_aac_a2_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a2_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"><SegmentTimeline><S d="96000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate></Representation>
          </AdaptationSet>
          <AdaptationSet lang="por" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a3_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a3_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"/>
            <Representation audioSamplingRate="48000" bandwidth="128000"
 codecs="mp4a.40.2" id="13"><SegmentTemplate
 initialization="EXAMPLE_PRODUCT_audio_aac_a3_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a3_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"><SegmentTimeline><S d="96000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate></Representation>
          </AdaptationSet>
          <AdaptationSet lang="spa" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="EXAMPLE_PRODUCT_audio_aac_a4_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a4_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"/>
```

```
            <Representation audioSamplingRate="48000" bandwidth="128000"
 codecs="mp4a.40.2" id="14"><SegmentTemplate
 initialization="EXAMPLE_PRODUCT_audio_aac_a4_128kinit.mp4"
 media="EXAMPLE_PRODUCT_audio_aac_a4_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"><SegmentTimeline><S d="96000" r="11" t="0"/></SegmentTimeline></
SegmentTemplate></Representation>
          </AdaptationSet>
        </Period>
```

DASH manifest time signal example

DASH origin manifest example for time signal

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

```
  <Period start="PT346530.250S" id="178443" duration="PT61.561S">
         <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
           <Event  duration="5310000">
             <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="183003"
 tier="4095">
               <scte35:TimeSignal>
                 <scte35:SpliceTime ptsTime="3442857000"/>
               </scte35:TimeSignal>
               <scte35:SegmentationDescriptor segmentationEventId="1414668"
 segmentationEventCancelIndicator="false" segmentationDuration="8100000">
                 <scte35:DeliveryRestrictions webDeliveryAllowedFlag="false"
 noRegionalBlackoutFlag="false" archiveAllowedFlag="false" deviceRestrictions="3"/>
                 <scte35:SegmentationUpid segmentationUpidType="12"
 segmentationUpidLength="2" segmentationTypeId="52" segmentNum="0"
 segmentsExpected="0">0100</scte35:SegmentationUpid>
               </scte35:SegmentationDescriptor>
             </scte35:SpliceInfoSection>
           </Event>
         </EventStream>
         <AdaptationSet mimeType="video/mp4" segmentAlignment="true"
 subsegmentAlignment="true" startWithSAP="1" subsegmentStartsWithSAP="1"
 bitstreamSwitching="true">
           <Representation id="1" width="960" height="540" frameRate="30000/1001"
 bandwidth="1000000" codecs="avc1.4D401F">
```

```
              <SegmentTemplate timescale="30000" media="index_video_1_0_$Number
$.mp4?m=1528475245" initialization="index_video_1_0_init.mp4?m=1528475245"
 startNumber="178444" presentationTimeOffset="10395907501">
                <SegmentTimeline>
                  <S t="10395907501" d="60060" r="29"/>
                  <S t="10397709301" d="45045"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet mimeType="audio/mp4" segmentAlignment="0" lang="eng">
            <Representation id="2" bandwidth="96964" audioSamplingRate="48000"
 codecs="mp4a.40.2">
              <SegmentTemplate timescale="48000" media="index_audio_2_0_$Number
$.mp4?m=1528475245" initialization="index_audio_2_0_init.mp4?m=1528475245"
 startNumber="178444" presentationTimeOffset="16633452001">
                <SegmentTimeline>
                  <S t="16633452289" d="96256" r="3"/>
                  <S t="16633837313" d="95232"/>
                  <S t="16633932545" d="96256" r="4"/>
                  <S t="16634413825" d="95232"/>
                  <S t="16634509057" d="96256" r="5"/>
                  <S t="16635086593" d="95232"/>
                  <S t="16635181825" d="96256" r="4"/>
                  <S t="16635663105" d="95232"/>
                  <S t="16635758337" d="96256" r="5"/>
                  <S t="16636335873" d="71680"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
        </Period>
```

DASH personalized response example for time signal

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

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

```
  <Period id="178443_1" start="PT96H15M30.25S">
```

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

```
            </Representation>
            <Representation bandwidth="1350000" codecs="avc1.64001e" height="396"
id="5" width="704">
              <SegmentTemplate initialization="nbc_fallback_ad_2_396p_6init.mp4"
media="nbc_fallback_ad_2_396p_6_$Number%09d$.mp4" startNumber="1" timescale="90000">
                <SegmentTimeline>
                  <S d="360000" r="6" t="0"/>
                  <S d="176940" t="2520000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="900000" codecs="avc1.64001e" height="396" id="6"
width="704">
              <SegmentTemplate initialization="nbc_fallback_ad_2_396p_5init.mp4"
media="nbc_fallback_ad_2_396p_5_$Number%09d$.mp4" startNumber="1" timescale="90000">
                <SegmentTimeline>
                  <S d="360000" r="6" t="0"/>
                  <S d="176940" t="2520000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="600000" codecs="avc1.64001e" height="396" id="7"
width="704">
              <SegmentTemplate initialization="nbc_fallback_ad_2_396p_4init.mp4"
media="nbc_fallback_ad_2_396p_4_$Number%09d$.mp4" startNumber="1" timescale="90000">
                <SegmentTimeline>
                  <S d="360000" r="6" t="0"/>
                  <S d="176940" t="2520000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="450000" codecs="avc1.640016" height="288" id="8"
width="512">
              <SegmentTemplate initialization="nbc_fallback_ad_2_288p_3init.mp4"
media="nbc_fallback_ad_2_288p_3_$Number%09d$.mp4" startNumber="1" timescale="90000">
                <SegmentTimeline>
                  <S d="360000" r="6" t="0"/>
                  <S d="176940" t="2520000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="300000" codecs="avc1.640016" height="288" id="9"
width="512">
```

```
            <SegmentTemplate initialization="nbc_fallback_ad_2_288p_2init.mp4"
media="nbc_fallback_ad_2_288p_2_$Number%09d$.mp4" startNumber="1" timescale="90000">
              <SegmentTimeline>
                <S d="360000" r="6" t="0"/>
                <S d="176940" t="2520000"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
          <Representation bandwidth="200000" codecs="avc1.640016" height="288"
id="10" width="512">
            <SegmentTemplate initialization="nbc_fallback_ad_2_288p_1init.mp4"
media="nbc_fallback_ad_2_288p_1_$Number%09d$.mp4" startNumber="1" timescale="90000">
              <SegmentTimeline>
                <S d="180000" r="13" t="0"/>
                <S d="176940" t="2520000"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
        </AdaptationSet>
        <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
          <SegmentTemplate
initialization="nbc_fallback_ad_2_audio_aac_a1_128kinit.mp4"
media="nbc_fallback_ad_2_audio_aac_a1_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000"/>
          <Representation audioSamplingRate="48000" bandwidth="128000"
codecs="mp4a.40.2" id="11">
            <SegmentTemplate
initialization="nbc_fallback_ad_2_audio_aac_a1_128kinit.mp4"
media="nbc_fallback_ad_2_audio_aac_a1_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000">
              <SegmentTimeline>
                <S d="96000" r="13" t="0"/>
                <S d="94368" t="1344000"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
        </AdaptationSet>
        <AdaptationSet lang="enm" mimeType="audio/mp4" segmentAlignment="0">
          <SegmentTemplate
initialization="nbc_fallback_ad_2_audio_aac_a2_128kinit.mp4"
media="nbc_fallback_ad_2_audio_aac_a2_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000"/>
          <Representation audioSamplingRate="48000" bandwidth="128000"
codecs="mp4a.40.2" id="12">
```

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

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

DASH manifest Base64-encoded binary example with single-period input

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

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

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

```
    <Period id="1" start="PT0S">
              <BaseURL>dash/</BaseURL>
              <EventStream schemeIdUri="urn:scte:scte35:2014:xml+bin" timescale="1">
                  <Event presentationTime="1550252760" duration="24" id="136">
                      <Signal xmlns="http://www.scte.org/schemas/35/2016">
                          <Binary>/DAhAAAAAAAAAP/wEAUAAACIf+9/fgAg9YDAAAAAAABiJjIs</
Binary>
                      </Signal>
                  </Event>
                  <Event presentationTime="1550252880" duration="24" id="137">
                      <Signal xmlns="http://www.scte.org/schemas/35/2016">
                          <Binary>/DAhAAAAAAAAAP/wEAUAAACJf+9/fgAg9YDAAAAAAC/KdNe</
Binary>
                      </Signal>
                  </Event>
                  <Event presentationTime="1550253000" duration="24" id="138">
                      <Signal xmlns="http://www.scte.org/schemas/35/2016">
                          <Binary>/DAhAAAAAAAAAP/wEAUAAACKf+9/fgAg9YDAAAAAADc+O1/</
Binary>
                      </Signal>
                  </Event>
              </EventStream>
              <AdaptationSet...
              </AdaptationSet>
          </Period>
```

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

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

```
   <Period id="0.0" start="PT0S">
              <BaseURL>dash/</BaseURL>
              <AdaptationSet audioSamplingRate="48000" codecs="mp4a.40.2"
 contentType="audio" group="1" id="1" mimeType="audio/mp4" segmentAlignment="true"
 startWithSAP="1">
                  <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="1"/>
                  <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
                  <Representation bandwidth="69000" id="audio=69000">
                      <SegmentTemplate initialization="scte35-$RepresentationID
$.dash" media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="0"
 startNumber="1" timescale="48000">
                          <SegmentTimeline>
                              <S d="48129" t="74412130844415"/>
                              <S d="48128" t="74412130892544"/>
                              <S d="48127" t="74412130940672"/>
                              <S d="48129" t="74412130988799"/>
                              <S d="48128" t="74412131036928"/>
                              <S d="47104" t="74412131085056"/>
                              <S d="48128" t="74412131132160"/>
                              <S d="48127" t="74412131180288"/>
                              <S d="48129" t="74412131228415"/>
                              <S d="48128" t="74412131276544"/>
                              <S d="48127" t="74412131324672"/>
                              <S d="48129" t="74412131372799"/>
                              <S d="48128" t="74412131420928"/>
                              <S d="47104" t="74412131469056"/>
                              <S d="48128" t="74412131516160"/>
                              <S d="48127" t="74412131564288"/>
                              <S d="48129" t="74412131612415"/>
                              <S d="48128" t="74412131660544"/>
                              <S d="48127" t="74412131708672"/>
                              <S d="48129" t="74412131756799"/>
                              <S d="48128" t="74412131804928"/>
```

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

```
media="visitalps_1080p30_video_1080p_10_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="180000" r="6" t="0"/>
                            <S d="86940" t="1260000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="3000000" codecs="avc1.64001f" height="720"
id="2" width="1280">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_720p_9init.mp4"
media="visitalps_1080p30_video_720p_9_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="180000" r="6" t="0"/>
                            <S d="86940" t="1260000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="1875000" codecs="avc1.64001f" height="720"
id="3" width="1280">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_720p_8init.mp4"
media="visitalps_1080p30_video_720p_8_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="270000" r="3" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="1500000" codecs="avc1.64001f" height="540"
id="4" width="960">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_540p_7init.mp4"
media="visitalps_1080p30_video_540p_7_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
```

```
                <Representation bandwidth="1012500" codecs="avc1.64001e" height="396"
id="5" width="704">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_396p_6init.mp4"
media="visitalps_1080p30_video_396p_6_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="675000" codecs="avc1.64001e" height="396"
id="6" width="704">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_396p_5init.mp4"
media="visitalps_1080p30_video_396p_5_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="450000" codecs="avc1.64001e" height="396"
id="7" width="704">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_396p_4init.mp4"
media="visitalps_1080p30_video_396p_4_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="337500" codecs="avc1.640016" height="288"
id="8" width="512">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_288p_3init.mp4"
media="visitalps_1080p30_video_288p_3_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
```

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

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

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

```
                    <S d="48128" t="74412134588160"/>
                    <S d="48127" t="74412134636288"/>
                    <S d="48129" t="74412134684415"/>
                    <S d="48128" t="74412134732544"/>
                    <S d="48127" t="74412134780672"/>
                    <S d="48129" t="74412134828799"/>
                    <S d="48128" t="74412134876928"/>
                    <S d="47104" t="74412134925056"/>
                    <S d="48128" t="74412134972160"/>
                    <S d="48127" t="74412135020288"/>
                    <S d="48129" t="74412135068415"/>
                    <S d="48128" t="74412135116544"/>
                    <S d="48127" t="74412135164672"/>
                    <S d="48129" t="74412135212799"/>
                    <S d="48128" t="74412135260928"/>
                    <S d="47104" t="74412135309056"/>
                    <S d="48128" t="74412135356160"/>
                    <S d="48127" t="74412135404288"/>
                    <S d="48129" t="74412135452415"/>
                    <S d="48128" t="74412135500544"/>
                    <S d="48127" t="74412135548672"/>
                    <S d="48129" t="74412135596799"/>
                    <S d="48128" t="74412135644928"/>
                    <S d="47104" t="74412135693056"/>
                    <S d="48128" t="74412135740160"/>
                    <S d="48127" t="74412135788288"/>
                    <S d="48129" t="74412135836415"/>
                    <S d="48128" t="74412135884544"/>
                    <S d="48127" t="74412135932672"/>
                    <S d="48129" t="74412135980799"/>
                    <S d="48128" t="74412136028928"/>
                    <S d="47104" t="74412136077056"/>
                    <S d="48128" t="74412136124160"/>
                    <S d="48127" t="74412136172288"/>
                    <S d="48129" t="74412136220415"/>
                    <S d="48128" t="74412136268544"/>
                    <S d="48127" t="74412136316672"/>
                    <S d="48129" t="74412136364799"/>
                    <S d="48128" t="74412136412928"/>
                    <S d="47104" t="74412136461056"/>
                    <S d="48128" t="74412136508160"/>
                    <S d="48127" t="74412136556288"/>
                    <S d="48129" t="74412136604415"/>
                    <S d="48128" t="74412136652544"/>
```

```
                                <S d="48127" t="74412136700672"/>
                                <S d="48129" t="74412136748799"/>
                                <S d="48128" t="74412136796928"/>
                                <S d="47104" t="74412136845056"/>
                                <S d="48128" t="74412136892160"/>
                                <S d="48127" t="74412136940288"/>
                                <S d="48129" t="74412136988415"/>
                                <S d="48128" t="74412137036544"/>
                                <S d="48127" t="74412137084672"/>
                                <S d="48129" t="74412137132799"/>
                                <S d="48128" t="74412137180928"/>
                                <S d="47104" t="74412137229056"/>
                                <S d="48128" t="74412137276160"/>
                                <S d="48127" t="74412137324288"/>
                                <S d="48129" t="74412137372415"/>
                                <S d="48128" t="74412137420544"/>
                                <S d="48127" t="74412137468672"/>
                                <S d="48129" t="74412137516799"/>
                                <S d="48128" t="74412137564928"/>
                                <S d="47104" t="74412137613056"/>
                                <S d="48128" t="74412137660160"/>
                                <S d="48127" t="74412137708288"/>
                                <S d="48129" t="74412137756415"/>
                                <S d="48128" t="74412137804544"/>
                                <S d="48127" t="74412137852672"/>
                                <S d="48129" t="74412137900799"/>
                                <S d="48128" t="74412137948928"/>
                                <S d="47104" t="74412137997056"/>
                                <S d="48128" t="74412138044160"/>
                                <S d="48127" t="74412138092288"/>
                                <S d="48129" t="74412138140415"/>
                                <S d="48128" t="74412138188544"/>
                                <S d="48127" t="74412138236672"/>
                            </SegmentTimeline>
                        </SegmentTemplate>
                    </Representation>
                </AdaptationSet>
                <AdaptationSet codecs="avc1.64001F" contentType="video" group="2"
height="720" id="2" mimeType="video/mp4" par="16:9" sar="1:1" segmentAlignment="true"
startWithSAP="1" width="1280">
                    <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
                    <Representation bandwidth="700000" id="video=700000"
scanType="progressive">
```

```
                    <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="139522750560000"
startNumber="60" timescale="90000">
                        <SegmentTimeline>
                            <S d="90000" r="95" t="139522750560000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
            </AdaptationSet>
        </Period>
        <Period id="1550252880.0_1" start="PT430625H48M">
            <BaseURL>http://d2gh0tfpz97e4o.cloudfront.net/visitalps/</BaseURL>
            <AdaptationSet bitstreamSwitching="false" frameRate="30/1"
mimeType="video/mp4" segmentAlignment="true" startWithSAP="1"
subsegmentAlignment="true" subsegmentStartsWithSAP="1">
                <SegmentTemplate startNumber="1" timescale="90000"/>
                <Representation bandwidth="7500000" codecs="avc1.640028"
height="1080" id="1" width="1920">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_1080p_10init.mp4"
media="visitalps_1080p30_video_1080p_10_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="180000" r="6" t="0"/>
                            <S d="86940" t="1260000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="3000000" codecs="avc1.64001f" height="720"
id="2" width="1280">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_720p_9init.mp4"
media="visitalps_1080p30_video_720p_9_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="180000" r="6" t="0"/>
                            <S d="86940" t="1260000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="1875000" codecs="avc1.64001f" height="720"
id="3" width="1280">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_720p_8init.mp4"
```

```
media="visitalps_1080p30_video_720p_8_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="270000" r="3" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="1500000" codecs="avc1.64001f" height="540"
id="4" width="960">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_540p_7init.mp4"
media="visitalps_1080p30_video_540p_7_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="1012500" codecs="avc1.64001e" height="396"
id="5" width="704">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_396p_6init.mp4"
media="visitalps_1080p30_video_396p_6_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="675000" codecs="avc1.64001e" height="396"
id="6" width="704">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_396p_5init.mp4"
media="visitalps_1080p30_video_396p_5_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
```

```
                <Representation bandwidth="450000" codecs="avc1.64001e" height="396"
id="7" width="704">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_396p_4init.mp4"
media="visitalps_1080p30_video_396p_4_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="337500" codecs="avc1.640016" height="288"
id="8" width="512">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_288p_3init.mp4"
media="visitalps_1080p30_video_288p_3_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="225000" codecs="avc1.640016" height="288"
id="9" width="512">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_288p_2init.mp4"
media="visitalps_1080p30_video_288p_2_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="150000" codecs="avc1.640016" height="288"
id="10" width="512">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_288p_1init.mp4"
media="visitalps_1080p30_video_288p_1_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="180000" r="6" t="0"/>
```

```
                            <S d="86940" t="1260000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
            </AdaptationSet>
            <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
                <SegmentTemplate
 initialization="visitalps_1080p30_audio_aac_128kinit.mp4"
 media="visitalps_1080p30_audio_aac_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000"/>
                <Representation audioSamplingRate="48000" bandwidth="128000"
 codecs="mp4a.40.2" id="11">
                    <SegmentTemplate
 initialization="visitalps_1080p30_audio_aac_128kinit.mp4"
 media="visitalps_1080p30_audio_aac_128k_$Number%09d$.mp4" startNumber="1"
 timescale="48000">
                        <SegmentTimeline>
                            <S d="96000" r="6" t="0"/>
                            <S d="46368" t="672000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
            </AdaptationSet>
        </Period>
        <Period id="1550252880.0" start="PT430625H48M14.966S">
            <BaseURL>dash/</BaseURL>
            <EventStream schemeIdUri="urn:scte:scte35:2014:xml+bin" timescale="1">
                <Event duration="24" id="137" presentationTime="1550252880">
                    <Signal xmlns="http://www.scte.org/schemas/35/2016">
                        <Binary>/DAhAAAAAAAAAP/wEAUAAACJf+9/fgAg9YDAAAAAAAC/KdNe</
Binary>
                    </Signal>
                </Event>
            </EventStream>
            <AdaptationSet audioSamplingRate="48000" codecs="mp4a.40.2"
 contentType="audio" group="1" id="1" mimeType="audio/mp4" segmentAlignment="true"
 startWithSAP="1">
                <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="1"/>
                <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
                <Representation bandwidth="69000" id="audio=69000">
                    <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
 media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="74412138958368"
 timescale="48000">
```

```
                        <SegmentTimeline>
                            <S d="48128" t="74412138956544"/>
                            <S d="48127" t="74412139004672"/>
                            <S d="48129" t="74412139052799"/>
                            <S d="48128" t="74412139100928"/>
                            <S d="47104" t="74412139149056"/>
                            <S d="48128" t="74412139196160"/>
                            <S d="48127" t="74412139244288"/>
                            <S d="48129" t="74412139292415"/>
                            <S d="48128" t="74412139340544"/>
                            <S d="48127" t="74412139388672"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
            </AdaptationSet>
            <AdaptationSet codecs="avc1.64001F" contentType="video" group="2"
height="720" id="2" mimeType="video/mp4" par="16:9" sar="1:1" segmentAlignment="true"
startWithSAP="1" width="1280">
                <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
                <Representation bandwidth="700000" id="video=700000"
scanType="progressive">
                    <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="139522760546940"
timescale="90000">
                        <SegmentTimeline>
                            <S d="90000" r="9" t="139522760460000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
            </AdaptationSet>
        </Period>
        <Period id="1550252904.0" start="PT430625H48M24S">
            <BaseURL>dash/</BaseURL>
            <AdaptationSet audioSamplingRate="48000" codecs="mp4a.40.2"
contentType="audio" group="1" id="1" mimeType="audio/mp4" segmentAlignment="true"
startWithSAP="1">
                <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="1"/>
                <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
                <Representation bandwidth="69000" id="audio=69000">
                    <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="74412139392000"
startNumber="180" timescale="48000">
                        <SegmentTimeline>
```

```
                    <S d="48129" t="74412139436799"/>
                    <S d="48128" t="74412139484928"/>
                    <S d="47104" t="74412139533056"/>
                    <S d="48128" t="74412139580160"/>
                    <S d="48127" t="74412139628288"/>
                    <S d="48129" t="74412139676415"/>
                    <S d="48128" t="74412139724544"/>
                    <S d="48127" t="74412139772672"/>
                    <S d="48129" t="74412139820799"/>
                    <S d="48128" t="74412139868928"/>
                    <S d="47104" t="74412139917056"/>
                    <S d="48128" t="74412139964160"/>
                    <S d="48127" t="74412140012288"/>
                    <S d="48129" t="74412140060415"/>
                    <S d="48128" t="74412140108544"/>
                    <S d="48127" t="74412140156672"/>
                    <S d="48129" t="74412140204799"/>
                    <S d="48128" t="74412140252928"/>
                    <S d="47104" t="74412140301056"/>
                    <S d="48128" t="74412140348160"/>
                    <S d="48127" t="74412140396288"/>
                    <S d="48129" t="74412140444415"/>
                    <S d="48128" t="74412140492544"/>
                    <S d="48127" t="74412140540672"/>
                    <S d="48129" t="74412140588799"/>
                    <S d="48128" t="74412140636928"/>
                    <S d="47104" t="74412140685056"/>
                    <S d="48128" t="74412140732160"/>
                    <S d="48127" t="74412140780288"/>
                    <S d="48129" t="74412140828415"/>
                    <S d="48128" t="74412140876544"/>
                    <S d="48127" t="74412140924672"/>
                    <S d="48129" t="74412140972799"/>
                    <S d="48128" t="74412141020928"/>
                    <S d="47104" t="74412141069056"/>
                    <S d="48128" t="74412141116160"/>
                    <S d="48127" t="74412141164288"/>
                    <S d="48129" t="74412141212415"/>
                    <S d="48128" t="74412141260544"/>
                    <S d="48127" t="74412141308672"/>
                    <S d="48129" t="74412141356799"/>
                    <S d="48128" t="74412141404928"/>
                    <S d="47104" t="74412141453056"/>
                    <S d="48128" t="74412141500160"/>
```

```
                    <S d="48127" t="74412141548288"/>
                    <S d="48129" t="74412141596415"/>
                    <S d="48128" t="74412141644544"/>
                    <S d="48127" t="74412141692672"/>
                    <S d="48129" t="74412141740799"/>
                    <S d="48128" t="74412141788928"/>
                    <S d="47104" t="74412141837056"/>
                    <S d="48128" t="74412141884160"/>
                    <S d="48127" t="74412141932288"/>
                    <S d="48129" t="74412141980415"/>
                    <S d="48128" t="74412142028544"/>
                    <S d="48127" t="74412142076672"/>
                    <S d="48129" t="74412142124799"/>
                    <S d="48128" t="74412142172928"/>
                    <S d="47104" t="74412142221056"/>
                    <S d="48128" t="74412142268160"/>
                    <S d="48127" t="74412142316288"/>
                    <S d="48129" t="74412142364415"/>
                    <S d="48128" t="74412142412544"/>
                    <S d="48127" t="74412142460672"/>
                    <S d="48129" t="74412142508799"/>
                    <S d="48128" t="74412142556928"/>
                    <S d="47104" t="74412142605056"/>
                    <S d="48128" t="74412142652160"/>
                    <S d="48127" t="74412142700288"/>
                    <S d="48129" t="74412142748415"/>
                    <S d="48128" t="74412142796544"/>
                    <S d="48127" t="74412142844672"/>
                    <S d="48129" t="74412142892799"/>
                    <S d="48128" t="74412142940928"/>
                    <S d="47104" t="74412142989056"/>
                    <S d="48128" t="74412143036160"/>
                    <S d="48127" t="74412143084288"/>
                    <S d="48129" t="74412143132415"/>
                    <S d="48128" t="74412143180544"/>
                    <S d="48127" t="74412143228672"/>
                    <S d="48129" t="74412143276799"/>
                    <S d="48128" t="74412143324928"/>
                    <S d="47104" t="74412143373056"/>
                    <S d="48128" t="74412143420160"/>
                    <S d="48127" t="74412143468288"/>
                    <S d="48129" t="74412143516415"/>
                    <S d="48128" t="74412143564544"/>
                    <S d="48127" t="74412143612672"/>
```

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

```
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="3000000" codecs="avc1.64001f" height="720"
id="2" width="1280">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_720p_9init.mp4"
media="visitalps_1080p30_video_720p_9_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="180000" r="6" t="0"/>
                            <S d="86940" t="1260000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="1875000" codecs="avc1.64001f" height="720"
id="3" width="1280">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_720p_8init.mp4"
media="visitalps_1080p30_video_720p_8_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="270000" r="3" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="1500000" codecs="avc1.64001f" height="540"
id="4" width="960">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_540p_7init.mp4"
media="visitalps_1080p30_video_540p_7_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="1012500" codecs="avc1.64001e" height="396"
id="5" width="704">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_396p_6init.mp4"
media="visitalps_1080p30_video_396p_6_$Number%09d$.mp4" startNumber="1"
timescale="90000">
```

```
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="675000" codecs="avc1.64001e" height="396"
id="6" width="704">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_396p_5init.mp4"
media="visitalps_1080p30_video_396p_5_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="450000" codecs="avc1.64001e" height="396"
id="7" width="704">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_396p_4init.mp4"
media="visitalps_1080p30_video_396p_4_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="337500" codecs="avc1.640016" height="288"
id="8" width="512">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_288p_3init.mp4"
media="visitalps_1080p30_video_288p_3_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="225000" codecs="avc1.640016" height="288"
id="9" width="512">
```

```
                    <SegmentTemplate
initialization="visitalps_1080p30_video_288p_2init.mp4"
media="visitalps_1080p30_video_288p_2_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="360000" r="2" t="0"/>
                            <S d="266940" t="1080000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation bandwidth="150000" codecs="avc1.640016" height="288"
id="10" width="512">
                    <SegmentTemplate
initialization="visitalps_1080p30_video_288p_1init.mp4"
media="visitalps_1080p30_video_288p_1_$Number%09d$.mp4" startNumber="1"
timescale="90000">
                        <SegmentTimeline>
                            <S d="180000" r="6" t="0"/>
                            <S d="86940" t="1260000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
            </AdaptationSet>
            <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
                <SegmentTemplate
initialization="visitalps_1080p30_audio_aac_128kinit.mp4"
media="visitalps_1080p30_audio_aac_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000"/>
                <Representation audioSamplingRate="48000" bandwidth="128000"
codecs="mp4a.40.2" id="11">
                    <SegmentTemplate
initialization="visitalps_1080p30_audio_aac_128kinit.mp4"
media="visitalps_1080p30_audio_aac_128k_$Number%09d$.mp4" startNumber="1"
timescale="48000">
                        <SegmentTimeline>
                            <S d="96000" r="6" t="0"/>
                            <S d="46368" t="672000"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
            </AdaptationSet>
        </Period>
        <Period id="1550253000.0" start="PT430625H50M14.966S">
            <BaseURL>dash/</BaseURL>
```

```
            <EventStream schemeIdUri="urn:scte:scte35:2014:xml+bin" timescale="1">
                <Event duration="24" id="138" presentationTime="1550253000">
                    <Signal xmlns="http://www.scte.org/schemas/35/2016">
                        <Binary>/DAhAAAAAAAAAP/wEAUAAACKf+9/fgAg9YDAAAAAAADc+O1/</
Binary>
                    </Signal>
                </Event>
            </EventStream>
            <AdaptationSet audioSamplingRate="48000" codecs="mp4a.40.2"
 contentType="audio" group="1" id="1" mimeType="audio/mp4" segmentAlignment="true"
 startWithSAP="1">
                <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="1"/>
                <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
                <Representation bandwidth="69000" id="audio=69000">
                    <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
 media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="74412144718368"
 timescale="48000">
                        <SegmentTimeline>
                            <S d="48128" t="74412144716544"/>
                            <S d="48127" t="74412144764672"/>
                            <S d="48129" t="74412144812799"/>
                            <S d="48128" t="74412144860928"/>
                            <S d="47104" t="74412144909056"/>
                            <S d="48128" t="74412144956160"/>
                            <S d="48127" t="74412145004288"/>
                            <S d="48129" t="74412145052415"/>
                            <S d="48128" t="74412145100544"/>
                            <S d="48127" t="74412145148672"/>
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
            </AdaptationSet>
            <AdaptationSet codecs="avc1.64001F" contentType="video" group="2"
 height="720" id="2" mimeType="video/mp4" par="16:9" sar="1:1" segmentAlignment="true"
 startWithSAP="1" width="1280">
                <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
                <Representation bandwidth="700000" id="video=700000"
 scanType="progressive">
                    <SegmentTemplate initialization="scte35-$RepresentationID$.dash"
 media="scte35-$RepresentationID$-$Time$.dash" presentationTimeOffset="139522771346940"
 timescale="90000">
                        <SegmentTimeline>
                            <S d="90000" r="9" t="139522771260000"/>
```

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

VOD DASH manifest examples

This section provides examples of VOD DASH manifests. Each example lists a manifest as received from the origin server and after MediaTailor has personalized the manifest with ads.

DASH VOD origin manifest

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

```
<Period start="PT0.000S" id="8778696" duration="PT29.229S">
          <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
 value="2018-07-27T09:35:44.011Z"></SupplementalProperty>
          <AdaptationSet mimeType="video/mp4" segmentAlignment="true" startWithSAP="1"
 subsegmentAlignment="true" subsegmentStartsWithSAP="1" bitstreamSwitching="true">
            <Representation id="1" width="960" height="540" frameRate="30000/1001"
 bandwidth="2200000" codecs="avc1.640029">
              <SegmentTemplate timescale="30000" media="index_video_7_0_$Number
$.mp4?m=1566416213" initialization="index_video_7_0_init.mp4?m=1566416213"
 startNumber="8778700" presentationTimeOffset="1317997547283">
                <SegmentTimeline>
                  <S t="1317997547283" d="180180" r="3"/>
                  <S t="1317998268003" d="156156"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation id="2" width="1280" height="720" frameRate="30000/1001"
 bandwidth="3299968" codecs="avc1.640029">
              <SegmentTemplate timescale="30000" media="index_video_10_0_$Number
$.mp4?m=1566416213" initialization="index_video_10_0_init.mp4?m=1566416213"
 startNumber="8778700" presentationTimeOffset="1317997547283">
                <SegmentTimeline>
                  <S t="1317997547283" d="180180" r="3"/>
                  <S t="1317998268003" d="156156"/>
                </SegmentTimeline>
              </SegmentTemplate>
```

```
            </Representation>
            <Representation id="3" width="640" height="360" frameRate="30000/1001"
 bandwidth="800000" codecs="avc1.4D401E">
              <SegmentTemplate timescale="30000" media="index_video_28_0_$Number
$.mp4?m=1566416213" initialization="index_video_28_0_init.mp4?m=1566416213"
 startNumber="8778700" presentationTimeOffset="1317997547283">
                <SegmentTimeline>
                  <S t="1317997547283" d="180180" r="3"/>
                  <S t="1317998268003" d="156156"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet mimeType="audio/mp4" segmentAlignment="0" lang="eng">
            <Label>eng</Label>
            <Representation id="4" bandwidth="96636" audioSamplingRate="48000"
 codecs="mp4a.40.2">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"></
AudioChannelConfiguration>
              <SegmentTemplate timescale="48000" media="index_audio_5_0_$Number
$.mp4?m=1566416213" initialization="index_audio_5_0_init.mp4?m=1566416213"
 startNumber="8778700" presentationTimeOffset="2108796075909">
                <SegmentTimeline>
                  <S t="2108796075909" d="288768"/>
                  <S t="2108796364677" d="287744"/>
                  <S t="2108796652421" d="288768"/>
                  <S t="2108796941189" d="287744"/>
                  <S t="2108797228933" d="249856"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation id="5" bandwidth="96636" audioSamplingRate="48000"
 codecs="mp4a.40.2">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"></
AudioChannelConfiguration>
              <SegmentTemplate timescale="48000" media="index_audio_8_0_$Number
$.mp4?m=1566416213" initialization="index_audio_8_0_init.mp4?m=1566416213"
 startNumber="8778700" presentationTimeOffset="2108796075909">
                <SegmentTimeline>
                  <S t="2108796075909" d="288768"/>
                  <S t="2108796364677" d="287744"/>
                  <S t="2108796652421" d="288768"/>
```

```
                <S t="2108796941189" d="287744"/>
                <S t="2108797228933" d="249856"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
          <Representation id="6" bandwidth="64643" audioSamplingRate="48000"
 codecs="mp4a.40.2">
            <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"></
AudioChannelConfiguration>
            <SegmentTemplate timescale="48000" media="index_audio_26_0_$Number
$.mp4?m=1566416213" initialization="index_audio_26_0_init.mp4?m=1566416213"
 startNumber="8778700" presentationTimeOffset="2108796075909">
              <SegmentTimeline>
                <S t="2108796075909" d="288768"/>
                <S t="2108796364677" d="287744"/>
                <S t="2108796652421" d="288768"/>
                <S t="2108796941189" d="287744"/>
                <S t="2108797228933" d="249856"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
        </AdaptationSet>
        <AdaptationSet mimeType="application/mp4" codecs="stpp"
 segmentAlignment="true" startWithSAP="1" bitstreamSwitching="true" lang="eng">
          <Label>eng</Label>
          <Representation id="7" bandwidth="0">
            <SegmentTemplate timescale="90000" media="index_subtitles_4_0_$Number
$.mp4?m=1566416213" initialization="index_subtitles_4_0_init.mp4?m=1566416213"
 startNumber="8778700" presentationTimeOffset="3953992641850">
              <SegmentTimeline>
                <S t="3953992641850" d="540540" r="3"/>
                <S t="3953994804010" d="468468"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
        </AdaptationSet>
      </Period>
      <Period start="PT29.229S" id="8778704" duration="PT18.818S">
        <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
 value="2018-07-27T09:36:13.240Z"></SupplementalProperty>
        <AdaptationSet mimeType="video/mp4" segmentAlignment="true" startWithSAP="1"
 subsegmentAlignment="true" subsegmentStartsWithSAP="1" bitstreamSwitching="true">
```

```
            <Representation id="1" width="960" height="540" frameRate="30000/1001"
 bandwidth="2200000" codecs="avc1.640029">
              <SegmentTemplate timescale="30000" media="index_video_7_0_$Number
$.mp4?m=1566416213" initialization="index_video_7_0_init.mp4?m=1566416213"
 startNumber="8778705" presentationTimeOffset="1317998424159">
                <SegmentTimeline>
                  <S t="1317998424159" d="24024"/>
                  <S t="1317998448183" d="180180" r="2"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation id="2" width="1280" height="720" frameRate="30000/1001"
 bandwidth="3299968" codecs="avc1.640029">
              <SegmentTemplate timescale="30000" media="index_video_10_0_$Number
$.mp4?m=1566416213" initialization="index_video_10_0_init.mp4?m=1566416213"
 startNumber="8778705" presentationTimeOffset="1317998424159">
                <SegmentTimeline>
                  <S t="1317998424159" d="24024"/>
                  <S t="1317998448183" d="180180" r="2"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation id="3" width="640" height="360" frameRate="30000/1001"
 bandwidth="800000" codecs="avc1.4D401E">
              <SegmentTemplate timescale="30000" media="index_video_28_0_$Number
$.mp4?m=1566416213" initialization="index_video_28_0_init.mp4?m=1566416213"
 startNumber="8778705" presentationTimeOffset="1317998424159">
                <SegmentTimeline>
                  <S t="1317998424159" d="24024"/>
                  <S t="1317998448183" d="180180" r="2"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet mimeType="audio/mp4" segmentAlignment="0" lang="eng">
            <Label>eng</Label>
            <Representation id="4" bandwidth="96636" audioSamplingRate="48000"
 codecs="mp4a.40.2">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"></
AudioChannelConfiguration>
              <SegmentTemplate timescale="48000" media="index_audio_5_0_$Number
$.mp4?m=1566416213" initialization="index_audio_5_0_init.mp4?m=1566416213"
 startNumber="8778705" presentationTimeOffset="2108797478789">
```

```
                <SegmentTimeline>
                  <S t="2108797478789" d="38912"/>
                  <S t="2108797517701" d="287744"/>
                  <S t="2108797805445" d="288768"/>
                  <S t="2108798094213" d="287744"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation id="5" bandwidth="96636" audioSamplingRate="48000"
 codecs="mp4a.40.2">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"></
AudioChannelConfiguration>
              <SegmentTemplate timescale="48000" media="index_audio_8_0_$Number
$.mp4?m=1566416213" initialization="index_audio_8_0_init.mp4?m=1566416213"
 startNumber="8778705" presentationTimeOffset="2108797478789">
                <SegmentTimeline>
                  <S t="2108797478789" d="38912"/>
                  <S t="2108797517701" d="287744"/>
                  <S t="2108797805445" d="288768"/>
                  <S t="2108798094213" d="287744"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation id="6" bandwidth="64643" audioSamplingRate="48000"
 codecs="mp4a.40.2">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"></
AudioChannelConfiguration>
              <SegmentTemplate timescale="48000" media="index_audio_26_0_$Number
$.mp4?m=1566416213" initialization="index_audio_26_0_init.mp4?m=1566416213"
 startNumber="8778705" presentationTimeOffset="2108797478789">
                <SegmentTimeline>
                  <S t="2108797478789" d="38912"/>
                  <S t="2108797517701" d="287744"/>
                  <S t="2108797805445" d="288768"/>
                  <S t="2108798094213" d="287744"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet mimeType="application/mp4" codecs="stpp"
 segmentAlignment="true" startWithSAP="1" bitstreamSwitching="true" lang="eng">
            <Label>eng</Label>
```

```
            <Representation id="7" bandwidth="0">
              <SegmentTemplate timescale="90000" media="index_subtitles_4_0_$Number
$.mp4?m=1566416213" initialization="index_subtitles_4_0_init.mp4?m=1566416213"
 startNumber="8778705" presentationTimeOffset="3953995272478">
                <SegmentTimeline>
                  <S t="3953995272478" d="72072"/>
                  <S t="3953995344550" d="540540" r="2"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
        </Period>
```

DASH VOD personalized response manifest

The following example reflects the personalization that MediaTailor applies to the origin manifest.

```
<?xml version="1.0" encoding="UTF-8"?>
      <MPD id="201" minBufferTime="PT30S" profiles="urn:mpeg:dash:profile:isoff-
main:2011" type="static" xmlns="urn:mpeg:dash:schema:mpd:2011"
 xmlns:scte35="urn:scte:scte35:2013:xml" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://
standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-
MPD.xsd"><BaseURL>https://123456789012.mediapackage.us-west-2.amazonaws.com/out/
v1/5f6a2197815e444a967f0c12f8325a11/</BaseURL>
        <Period duration="PT14.976S" id="8778696_PT0S_0"
 start="PT0S"><BaseURL>https://444455556666.mediatailor.us-west-2.amazonaws.com/v1/
dashsegment/0d598fad40f42c4644d1c5b7674438772ee23b12/dash-vod-insertion/a5a7cf24-
ee56-40e9-a0a2-82b483cf8650/8778696_PT0S/8778696_PT0S_0/</BaseURL>
          <AdaptationSet bitstreamSwitching="false" frameRate="30/1" mimeType="video/
mp4" segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
            <SegmentTemplate startNumber="1" timescale="90000"/>
            <Representation bandwidth="3296000" codecs="avc1.64001f" height="720"
 id="1" width="1280">
              <SegmentTemplate initialization="asset_720_3_1init.mp4"
 media="asset_720_3_1_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="180000" r="6" t="0"/>
                  <S d="87000" t="1260000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
```

```
            <Representation bandwidth="2200000" codecs="avc1.64001f" height="540"
id="2" width="960">
              <SegmentTemplate initialization="asset_540_2_0init.mp4"
media="asset_540_2_0_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="180000" r="6" t="0"/>
                  <S d="87000" t="1260000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="800000" codecs="avc1.64001e" height="360" id="3"
width="640">
              <SegmentTemplate initialization="asset_360_0_2init.mp4"
media="asset_360_0_2_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="180000" r="6" t="0"/>
                  <S d="87000" t="1260000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="asset_audio_96_3init.mp4"
media="asset_audio_96_3_$Number%09d$.mp4" startNumber="1" timescale="48000"/>
            <Label>eng</Label>
            <Representation audioSamplingRate="48000" bandwidth="96000"
codecs="mp4a.40.2" id="4">
              <SegmentTemplate initialization="asset_audio_96_3init.mp4"
media="asset_audio_96_3_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="96256" r="3" t="0"/>
                  <S d="95232" t="385024"/>
                  <S d="96256" r="1" t="480256"/>
                  <S d="46080" t="672768"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="asset_audio_96_4init.mp4"
media="asset_audio_96_4_$Number%09d$.mp4" startNumber="1" timescale="48000"/>
            <Label>eng</Label>
            <Representation audioSamplingRate="48000" bandwidth="96000"
codecs="mp4a.40.2" id="5">
```

```
              <SegmentTemplate initialization="asset_audio_96_4init.mp4"
 media="asset_audio_96_4_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="96256" r="3" t="0"/>
                  <S d="95232" t="385024"/>
                  <S d="96256" r="1" t="480256"/>
                  <S d="46080" t="672768"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
        </Period><Period duration="PT14.976S" id="8778696_PT0S_1"
 start="PT14.976S"><BaseURL>https://444455556666.mediatailor.us-west-2.amazonaws.com/
v1/dashsegment/0d598fad40f42c4644d1c5b7674438772ee23b12/dash-vod-insertion/a5a7cf24-
ee56-40e9-a0a2-82b483cf8650/8778696_PT0S/8778696_PT0S_1/</BaseURL>
          <AdaptationSet bitstreamSwitching="false" frameRate="30/1" mimeType="video/
mp4" segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
            <SegmentTemplate startNumber="1" timescale="90000"/>
            <Representation bandwidth="3296000" codecs="avc1.64001f" height="720"
 id="1" width="1280">
              <SegmentTemplate initialization="asset_720_3_1init.mp4"
 media="asset_720_3_1_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="180000" r="6" t="0"/>
                  <S d="87000" t="1260000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="2200000" codecs="avc1.64001f" height="540"
 id="2" width="960">
              <SegmentTemplate initialization="asset_540_2_0init.mp4"
 media="asset_540_2_0_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="180000" r="6" t="0"/>
                  <S d="87000" t="1260000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="800000" codecs="avc1.64001e" height="360" id="3"
 width="640">
              <SegmentTemplate initialization="asset_360_0_2init.mp4"
 media="asset_360_0_2_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
```

```
                <S d="180000" r="6" t="0"/>
                <S d="87000" t="1260000"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
        </AdaptationSet>
        <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
          <SegmentTemplate initialization="asset_audio_96_3init.mp4"
media="asset_audio_96_3_$Number%09d$.mp4" startNumber="1" timescale="48000"/>
          <Label>eng</Label>
          <Representation audioSamplingRate="48000" bandwidth="96000"
codecs="mp4a.40.2" id="4">
            <SegmentTemplate initialization="asset_audio_96_3init.mp4"
media="asset_audio_96_3_$Number%09d$.mp4" startNumber="1">
              <SegmentTimeline>
                <S d="96256" r="3" t="0"/>
                <S d="95232" t="385024"/>
                <S d="96256" r="1" t="480256"/>
                <S d="46080" t="672768"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
        </AdaptationSet>
        <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
          <SegmentTemplate initialization="asset_audio_96_4init.mp4"
media="asset_audio_96_4_$Number%09d$.mp4" startNumber="1" timescale="48000"/>
          <Label>eng</Label>
          <Representation audioSamplingRate="48000" bandwidth="96000"
codecs="mp4a.40.2" id="5">
            <SegmentTemplate initialization="asset_audio_96_4init.mp4"
media="asset_audio_96_4_$Number%09d$.mp4" startNumber="1">
              <SegmentTimeline>
                <S d="96256" r="3" t="0"/>
                <S d="95232" t="385024"/>
                <S d="96256" r="1" t="480256"/>
                <S d="46080" t="672768"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
        </AdaptationSet>
      </Period><Period duration="PT24.024S" id="8778696_PT29.952S" start="PT29.952S">
        <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
value="2018-07-27T09:35:44.011Z"/>
```

```
          <AdaptationSet bitstreamSwitching="true" mimeType="video/mp4"
 segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
            <Representation bandwidth="2200000" codecs="avc1.640029"
 frameRate="30000/1001" height="540" id="1" width="960">
              <SegmentTemplate initialization="index_video_7_0_init.mp4?
m=1566416213" media="index_video_7_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="1317997547283" startNumber="8778700" timescale="30000">
                <SegmentTimeline><S d="180180" r="3" t="1317997547283"/></
SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="3299968" codecs="avc1.640029"
 frameRate="30000/1001" height="720" id="2" width="1280">
              <SegmentTemplate initialization="index_video_10_0_init.mp4?
m=1566416213" media="index_video_10_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="1317997547283" startNumber="8778700" timescale="30000">
                <SegmentTimeline><S d="180180" r="3" t="1317997547283"/></
SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="800000" codecs="avc1.4D401E"
 frameRate="30000/1001" height="360" id="3" width="640">
              <SegmentTemplate initialization="index_video_28_0_init.mp4?
m=1566416213" media="index_video_28_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="1317997547283" startNumber="8778700" timescale="30000">
                <SegmentTimeline><S d="180180" r="3" t="1317997547283"/></
SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <Label>eng</Label>
            <Representation audioSamplingRate="48000" bandwidth="96636"
 codecs="mp4a.40.2" id="4">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
              <SegmentTemplate initialization="index_audio_5_0_init.mp4?
m=1566416213" media="index_audio_5_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="2108796075909" startNumber="8778700" timescale="48000">
                <SegmentTimeline><S d="288768" t="2108796075909"/><S d="287744"
 t="2108796364677"/><S d="288768" t="2108796652421"/><S d="287744" t="2108796941189"/
><S d="249856" t="2108797228933"/></SegmentTimeline>
              </SegmentTemplate>
```

```
            </Representation>
            <Representation audioSamplingRate="48000" bandwidth="96636"
 codecs="mp4a.40.2" id="5">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
              <SegmentTemplate initialization="index_audio_8_0_init.mp4?
m=1566416213" media="index_audio_8_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="2108796075909" startNumber="8778700" timescale="48000">
                <SegmentTimeline><S d="288768" t="2108796075909"/><S d="287744"
 t="2108796364677"/><S d="288768" t="2108796652421"/><S d="287744" t="2108796941189"/
><S d="249856" t="2108797228933"/></SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation audioSamplingRate="48000" bandwidth="64643"
 codecs="mp4a.40.2" id="6">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
              <SegmentTemplate initialization="index_audio_26_0_init.mp4?
m=1566416213" media="index_audio_26_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="2108796075909" startNumber="8778700" timescale="48000">
                <SegmentTimeline><S d="288768" t="2108796075909"/><S d="287744"
 t="2108796364677"/><S d="288768" t="2108796652421"/><S d="287744" t="2108796941189"/
><S d="249856" t="2108797228933"/></SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet bitstreamSwitching="true" codecs="stpp" lang="eng"
 mimeType="application/mp4" segmentAlignment="true" startWithSAP="1">
            <Label>eng</Label>
            <Representation bandwidth="0" id="7">
              <SegmentTemplate initialization="index_subtitles_4_0_init.mp4?
m=1566416213" media="index_subtitles_4_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="3953992641850" startNumber="8778700" timescale="90000">
                <SegmentTimeline><S d="540540" r="3" t="3953992641850"/></
SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
        </Period><Period duration="PT14.976S" id="8778696_PT25S_0"
 start="PT53.976S"><BaseURL>https://444455556666.mediatailor.us-west-2.amazonaws.com/
v1/dashsegment/0d598fad40f42c4644d1c5b7674438772ee23b12/dash-vod-insertion/a5a7cf24-
ee56-40e9-a0a2-82b483cf8650/8778696_PT25S/8778696_PT25S_0/</BaseURL>
```

```
          <AdaptationSet bitstreamSwitching="false" frameRate="30/1" mimeType="video/
mp4" segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
            <SegmentTemplate startNumber="1" timescale="90000"/>
            <Representation bandwidth="3296000" codecs="avc1.64001f" height="720"
 id="1" width="1280">
              <SegmentTemplate initialization="asset_720_3_1init.mp4"
 media="asset_720_3_1_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="180000" r="6" t="0"/>
                  <S d="87000" t="1260000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="2200000" codecs="avc1.64001f" height="540"
 id="2" width="960">
              <SegmentTemplate initialization="asset_540_2_0init.mp4"
 media="asset_540_2_0_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="180000" r="6" t="0"/>
                  <S d="87000" t="1260000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="800000" codecs="avc1.64001e" height="360" id="3"
 width="640">
              <SegmentTemplate initialization="asset_360_0_2init.mp4"
 media="asset_360_0_2_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="180000" r="6" t="0"/>
                  <S d="87000" t="1260000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="asset_audio_96_3init.mp4"
 media="asset_audio_96_3_$Number%09d$.mp4" startNumber="1" timescale="48000"/>
            <Label>eng</Label>
            <Representation audioSamplingRate="48000" bandwidth="96000"
 codecs="mp4a.40.2" id="4">
              <SegmentTemplate initialization="asset_audio_96_3init.mp4"
 media="asset_audio_96_3_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
```

```
                <S d="96256" r="3" t="0"/>
                <S d="95232" t="385024"/>
                <S d="96256" r="1" t="480256"/>
                <S d="46080" t="672768"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
        </AdaptationSet>
        <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
          <SegmentTemplate initialization="asset_audio_96_4init.mp4"
 media="asset_audio_96_4_$Number%09d$.mp4" startNumber="1" timescale="48000"/>
          <Label>eng</Label>
          <Representation audioSamplingRate="48000" bandwidth="96000"
 codecs="mp4a.40.2" id="5">
            <SegmentTemplate initialization="asset_audio_96_4init.mp4"
 media="asset_audio_96_4_$Number%09d$.mp4" startNumber="1">
              <SegmentTimeline>
                <S d="96256" r="3" t="0"/>
                <S d="95232" t="385024"/>
                <S d="96256" r="1" t="480256"/>
                <S d="46080" t="672768"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
        </AdaptationSet>
      </Period><Period duration="PT14.976S" id="8778696_PT25S_1"
 start="PT1M8.952S"><BaseURL>https://444455556666.mediatailor.us-west-2.amazonaws.com/
v1/dashsegment/0d598fad40f42c4644d1c5b7674438772ee23b12/dash-vod-insertion/a5a7cf24-
ee56-40e9-a0a2-82b483cf8650/8778696_PT25S/8778696_PT25S_1/</BaseURL>
        <AdaptationSet bitstreamSwitching="false" frameRate="30/1" mimeType="video/
mp4" segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
          <SegmentTemplate startNumber="1" timescale="90000"/>
          <Representation bandwidth="3296000" codecs="avc1.64001f" height="720"
 id="1" width="1280">
            <SegmentTemplate initialization="asset_720_3_1init.mp4"
 media="asset_720_3_1_$Number%09d$.mp4" startNumber="1">
              <SegmentTimeline>
                <S d="180000" r="6" t="0"/>
                <S d="87000" t="1260000"/>
              </SegmentTimeline>
            </SegmentTemplate>
          </Representation>
```

```
            <Representation bandwidth="2200000" codecs="avc1.64001f" height="540"
id="2" width="960">
              <SegmentTemplate initialization="asset_540_2_0init.mp4"
media="asset_540_2_0_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="180000" r="6" t="0"/>
                  <S d="87000" t="1260000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="800000" codecs="avc1.64001e" height="360" id="3"
width="640">
              <SegmentTemplate initialization="asset_360_0_2init.mp4"
media="asset_360_0_2_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="180000" r="6" t="0"/>
                  <S d="87000" t="1260000"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="asset_audio_96_3init.mp4"
media="asset_audio_96_3_$Number%09d$.mp4" startNumber="1" timescale="48000"/>
            <Label>eng</Label>
            <Representation audioSamplingRate="48000" bandwidth="96000"
codecs="mp4a.40.2" id="4">
              <SegmentTemplate initialization="asset_audio_96_3init.mp4"
media="asset_audio_96_3_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="96256" r="3" t="0"/>
                  <S d="95232" t="385024"/>
                  <S d="96256" r="1" t="480256"/>
                  <S d="46080" t="672768"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="asset_audio_96_4init.mp4"
media="asset_audio_96_4_$Number%09d$.mp4" startNumber="1" timescale="48000"/>
            <Label>eng</Label>
            <Representation audioSamplingRate="48000" bandwidth="96000"
codecs="mp4a.40.2" id="5">
```

```
              <SegmentTemplate initialization="asset_audio_96_4init.mp4"
 media="asset_audio_96_4_$Number%09d$.mp4" startNumber="1">
                <SegmentTimeline>
                  <S d="96256" r="3" t="0"/>
                  <S d="95232" t="385024"/>
                  <S d="96256" r="1" t="480256"/>
                  <S d="46080" t="672768"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
        </Period><Period duration="PT5.205S" id="8778696_PT1M23.928S"
 start="PT1M23.928S">
          <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
 value="2018-07-27T09:35:44.011Z"/>
          <AdaptationSet bitstreamSwitching="true" mimeType="video/mp4"
 segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
            <Representation bandwidth="2200000" codecs="avc1.640029"
 frameRate="30000/1001" height="540" id="1" width="960">
              <SegmentTemplate initialization="index_video_7_0_init.mp4?
m=1566416213" media="index_video_7_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="1317998268003" startNumber="8778704" timescale="30000">
                <SegmentTimeline><S d="156156" t="1317998268003"/></SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="3299968" codecs="avc1.640029"
 frameRate="30000/1001" height="720" id="2" width="1280">
              <SegmentTemplate initialization="index_video_10_0_init.mp4?
m=1566416213" media="index_video_10_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="1317998268003" startNumber="8778704" timescale="30000">
                <SegmentTimeline><S d="156156" t="1317998268003"/></SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="800000" codecs="avc1.4D401E"
 frameRate="30000/1001" height="360" id="3" width="640">
              <SegmentTemplate initialization="index_video_28_0_init.mp4?
m=1566416213" media="index_video_28_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="1317998268003" startNumber="8778704" timescale="30000">
                <SegmentTimeline><S d="156156" t="1317998268003"/></SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
```

```
            <Label>eng</Label>
            <Representation audioSamplingRate="48000" bandwidth="96636"
 codecs="mp4a.40.2" id="4">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
              <SegmentTemplate initialization="index_audio_5_0_init.mp4?
m=1566416213" media="index_audio_5_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="2108797229061" startNumber="8778704" timescale="48000">
                <SegmentTimeline><S d="249856" t="2108797228933"/></SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation audioSamplingRate="48000" bandwidth="96636"
 codecs="mp4a.40.2" id="5">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
              <SegmentTemplate initialization="index_audio_8_0_init.mp4?
m=1566416213" media="index_audio_8_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="2108797229061" startNumber="8778704" timescale="48000">
                <SegmentTimeline><S d="249856" t="2108797228933"/></SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation audioSamplingRate="48000" bandwidth="64643"
 codecs="mp4a.40.2" id="6">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
              <SegmentTemplate initialization="index_audio_26_0_init.mp4?
m=1566416213" media="index_audio_26_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="2108797229061" startNumber="8778704" timescale="48000">
                <SegmentTimeline><S d="249856" t="2108797228933"/></SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet bitstreamSwitching="true" codecs="stpp" lang="eng"
 mimeType="application/mp4" segmentAlignment="true" startWithSAP="1">
            <Label>eng</Label>
            <Representation bandwidth="0" id="7">
              <SegmentTemplate initialization="index_subtitles_4_0_init.mp4?
m=1566416213" media="index_subtitles_4_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="3953994804010" startNumber="8778704" timescale="90000">
                <SegmentTimeline><S d="468468" t="3953994804010"/></SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
        </Period><Period duration="PT18.818S" id="8778704" start="PT1M29.133S">
```

```
          <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
 value="2018-07-27T09:36:13.240Z"/>
          <AdaptationSet bitstreamSwitching="true" mimeType="video/mp4"
 segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
            <Representation bandwidth="2200000" codecs="avc1.640029"
 frameRate="30000/1001" height="540" id="1" width="960">
              <SegmentTemplate initialization="index_video_7_0_init.mp4?
m=1566416213" media="index_video_7_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="1317998424159" startNumber="8778705" timescale="30000">
                <SegmentTimeline>
                  <S d="24024" t="1317998424159"/>
                  <S d="180180" r="2" t="1317998448183"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="3299968" codecs="avc1.640029"
 frameRate="30000/1001" height="720" id="2" width="1280">
              <SegmentTemplate initialization="index_video_10_0_init.mp4?
m=1566416213" media="index_video_10_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="1317998424159" startNumber="8778705" timescale="30000">
                <SegmentTimeline>
                  <S d="24024" t="1317998424159"/>
                  <S d="180180" r="2" t="1317998448183"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation bandwidth="800000" codecs="avc1.4D401E"
 frameRate="30000/1001" height="360" id="3" width="640">
              <SegmentTemplate initialization="index_video_28_0_init.mp4?
m=1566416213" media="index_video_28_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="1317998424159" startNumber="8778705" timescale="30000">
                <SegmentTimeline>
                  <S d="24024" t="1317998424159"/>
                  <S d="180180" r="2" t="1317998448183"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
          <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <Label>eng</Label>
            <Representation audioSamplingRate="48000" bandwidth="96636"
 codecs="mp4a.40.2" id="4">
```

```
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
              <SegmentTemplate initialization="index_audio_5_0_init.mp4?
m=1566416213" media="index_audio_5_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="2108797478789" startNumber="8778705" timescale="48000">
                <SegmentTimeline>
                  <S d="38912" t="2108797478789"/>
                  <S d="287744" t="2108797517701"/>
                  <S d="288768" t="2108797805445"/>
                  <S d="287744" t="2108798094213"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation audioSamplingRate="48000" bandwidth="96636"
 codecs="mp4a.40.2" id="5">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
              <SegmentTemplate initialization="index_audio_8_0_init.mp4?
m=1566416213" media="index_audio_8_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="2108797478789" startNumber="8778705" timescale="48000">
                <SegmentTimeline>
                  <S d="38912" t="2108797478789"/>
                  <S d="287744" t="2108797517701"/>
                  <S d="288768" t="2108797805445"/>
                  <S d="287744" t="2108798094213"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
            <Representation audioSamplingRate="48000" bandwidth="64643"
 codecs="mp4a.40.2" id="6">
              <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
              <SegmentTemplate initialization="index_audio_26_0_init.mp4?
m=1566416213" media="index_audio_26_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="2108797478789" startNumber="8778705" timescale="48000">
                <SegmentTimeline>
                  <S d="38912" t="2108797478789"/>
                  <S d="287744" t="2108797517701"/>
                  <S d="288768" t="2108797805445"/>
                  <S d="287744" t="2108798094213"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
```

```
          <AdaptationSet bitstreamSwitching="true" codecs="stpp" lang="eng"
 mimeType="application/mp4" segmentAlignment="true" startWithSAP="1">
            <Label>eng</Label>
            <Representation bandwidth="0" id="7">
              <SegmentTemplate initialization="index_subtitles_4_0_init.mp4?
m=1566416213" media="index_subtitles_4_0_$Number$.mp4?m=1566416213"
 presentationTimeOffset="3953995272478" startNumber="8778705" timescale="90000">
                <SegmentTimeline>
                  <S d="72072" t="3953995272478"/>
                  <S d="540540" r="2" t="3953995344550"/>
                </SegmentTimeline>
              </SegmentTemplate>
            </Representation>
          </AdaptationSet>
        </Period></MPD>
```

DASH location feature

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

What is the location feature?

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

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

The DASH specification provides a solution to this problem in the location feature, which is enabled by default in AWS Elemental MediaTailor configurations. When this feature is enabled, MediaTailor

puts the absolute URL in the manifest <Location> tag. Players that don't support sticky HTTP redirects can use the URL provided in <Location> to request updates to the manifest.

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

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

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

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

Do I need to use the location feature?

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

Example

Example URLs and example <Location> tag.

- **Example Example: Initial request URL**

```
https://777788889999.mediatailor.us-east-1.amazonaws.com/v1/
dash/5ca4c1892b1f213a1247fad47b3e34c454a7d490/testLocationTag/index.mpd
```

- **Example Example: Redirected 302 response**

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

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

```
<Location>https://777788889999.mediatailor.us-east-1.amazonaws.com/v1/
dash/5ca4c1892b1f213a1247fad47b3e34c454a7d490/testLocationTag/index.mpd?
aws.sessionId=0e5d9b45-ae97-49eb-901b-893d043e0aa6</Location>
```

Securing AWS Elemental MediaTailor origin interactions with SigV4

Signature Version 4 (SigV4) is a signing protocol used to authenticate MediaTailor requests to supported origins over HTTPS. With SigV4 signing, MediaTailor includes a signed authorization header in the HTTPS origin request to MediaTailor Channel Assembly, Amazon S3 and AWS Elemental MediaPackage version 2.

You can use SigV4 at your origin to ensure that manifest requests are only fulfilled if they're from MediaTailor and contain a signed authorization header. This way, unauthorized MediaTailor playback configurations are blocked from accessing your origin content. If the signed authorization header is valid, your origin fulfills the request. If it isn't valid, the request fails.

The following sections describe requirements for using MediaTailor SigV4 signing to supported origins.

MediaTailor Channel Assembly requirements

If you use SigV4 to protect your MediaTailor Channel Assembly origin, the following requirements must be met for MediaTailor to access the manifest:

- The origin base URL in your MediaTailor configuration must be a Channel Assembly channel in the following format: `channel-assembly.mediatailor.region.amazonaws.com`
- Your origin must be configured to use HTTPS. If HTTPS is not enabled at the origin, MediaTailor will not sign the request.
- Your channel must have an origin access policy that includes the following:
  - Principal access for MediaTailor to access your channel. Grant access to **mediatailor.amazonaws.com**.
  - IAM permissions **mediatailor:GetManifest** to read all top-level manifests referenced by the MediaTailor configuration.

  For information about setting a policy on the channel, see Create a channel using the MediaTailor console.

Example origin access policy for Channel Assembly, scoped to the MediaTailor configuration account

```
{
    "Effect": "Allow",
    "Principal": {"Service": "mediatailor.amazonaws.com"},
    "Action": "mediatailor:GetManifest",
    "Resource": "arn:aws:mediatailor:us-west-2:777788889999:channel/ca-origin-channel",
    "Condition": {
        "StringEquals": {"AWS:SourceAccount": "777788889999"}
    }
}
```

Example origin access policy for Channel Assembly, scoped to the MediaTailor playback configuration

```
{
    "Effect": "Allow",
    "Principal": {"Service": "mediatailor.amazonaws.com"},
    "Action": "mediatailor:GetManifest",
    "Resource": "arn:aws:mediatailor:us-west-2:777788889999:channel/ca-origin-channel",
    "Condition": {
        "StringEquals": {"AWS:SourceArn": "arn:aws:mediatailor:us-
west-2:777788889999:playbackConfiguration/test"}
    }
}
```

Amazon S3 requirements

If you use SigV4 to protect your Amazon S3 origin, the following requirements must be met for MediaTailor to access the manifest:

- The origin base URL in your MediaTailor configuration must be an S3 bucket in the following format: `s3.region.amazonaws.com`
- Your origin must be configured to use HTTPS. If HTTPS is not enabled at the origin, MediaTailor will not sign the request.
- Your channel must have an origin access policy that includes the following:
  - Principal access for MediaTailor to access your bucket. Grant access to **mediatailor.amazonaws.com.**

For information about configuring access in IAM, see Access management in the *AWS Identity and Access Management User Guide*.

- IAM permissions **s3:GetObject** to read all top-level manifests referenced by the MediaTailor configuration.

For general information about SigV4 for Amazon S3, see the Authenticating Requests (AWS Signature Version 4) topic in the *Amazon S3 API reference*.

Example origin access policy for Amazon S3, scoped to the MediaTailor account

```
{
    "Effect": "Allow",
    "Principal": {"Service": "mediatailor.amazonaws.com"},
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::mybucket/*",
    "Condition": {
        "StringEquals": {"AWS:SourceAccount": "111122223333"}
    }
}
```

Example origin access policy for Amazon S3, scoped to the MediaTailor playback configuration

```
{
    "Effect": "Allow",
    "Principal": {"Service": "mediatailor.amazonaws.com"},
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::mybucket/*",
    "Condition": {
        "StringEquals": {"AWS:SourceArn": "arn:aws:mediatailor:us-
west-2:111122223333:playbackConfiguration/test”}
    }
}
```

MediaPackage requirements

If you use SigV4 to protect your MediaPackage v2 origin, the following requirements must be met for MediaTailor to access the manifest:

- The origin base URL in your MediaTailor configuration must be a MediaPackage v2 endpoint in the following format: `mediapackagev2.region.amazonaws.com`

- Your origin must be configured to use HTTPS. If HTTPS is not enabled at the origin, MediaTailor will not sign the request.
- Your channel must have an origin access policy that includes the following:
  - Principal access for MediaTailor to access your endpoint. Grant access to **mediatailor.amazonaws.com.**
  - IAM permissions **mediapackagev2:GetObject** to read all top-level manifests referenced by the MediaTailor configuration.

For general information about SigV4 for MediaPackage v2, see the Authenticating Requests (AWS Signature Version 4) topic in the *MediaPackage v2 API reference*.

Example origin access policy for MediaPackage v2, scoped to the MediaTailor account

```
{
    "Effect": "Allow",
    "Principal": {"Service": "mediatailor.amazonaws.com"},
    "Action": "mediapackagev2:GetObject",
    "Resource": "arn:aws:mediapackagev2:us-west-2:444455556666:channelGroup/emp-origin-
channel-group/channel/emp-origin-channel/originEndpoint/emp-origin-endpoint",
    "Condition": {
        "StringEquals": {"AWS:SourceAccount": "444455556666"}
    }
}
```

Example origin access policy for MediaPackage v2, scoped to the MediaTailor playback configuration

```
{
    "Effect": "Allow",
    "Principal": {"Service": "mediatailor.amazonaws.com"},
    "Action": "mediapackagev2:GetObject",
    "Resource": "arn:aws:mediapackagev2:us-west-2:444455556666:channelGroup/emp-origin-
channel-group/channel/emp-origin-channel/originEndpoint/emp-origin-endpoint",
    "Condition": {
        "StringEquals": {"AWS:SourceArn": "arn:aws:mediatailor:us-
west-2:444455556666:playbackConfiguration/test”"}
    }
}
```

Integrating AWS Elemental MediaTailor with Google Ad Manager

Integrate MediaTailor with Google Ad Manager (Ad Manager) for programmatic access to an online, auction-driven marketplace where ad impressions can be bought and sold in real time. You must have an account set up with Ad Manager, then you can integrate with Ad Manager in the following ways:

- A server-side integration using an SSL certificate.
- A client-side player integration using the Programmatic Access Libraries (PAL) SDK. This integration is required if you want to use the open auction transaction type.

Ad Manager support for programmatic transaction types varies based on the type of integration that you're using. For a list of available options, see Transaction types or contact your Google account team.

The following sections describe these integrations in detail.

Topics

- Server-side AWS Elemental MediaTailor integration with Google Ad Manager
- Client-side AWS Elemental MediaTailor integration with Google Ad Manager

Server-side AWS Elemental MediaTailor integration with Google Ad Manager

Server-side ad requests to Google Ad Manager (Ad Manager) must include the SSL certificate that Ad Manager has issued to MediaTailor to authorize programmatic transactions.

To make server-side ad requests to Ad Manager

1. Submit an AWS Support ticket to request SSL certificates to be enabled. Include the following information in the Support ticket:
   - AWS Region
   - AWS account ID
   - MediaTailor playback configuration name

If you don't enable SSL certificates, Ad Manager responds to MediaTailor ad requests with HTTP 401 error codes in the `ERROR_ADS_INVALID_RESPONSE` ADS interaction log event type.

2. After SSL certificates are enabled, update the URL and parameters for your ADS and preroll ADS in the playback configuration. To update or create a playback configuration, see Working with AWS Elemental MediaTailor playback configurations.

   For official guidance on VAST ad request URL parameters for Ad Manager, see the Ad Manager Server-side implementation guide. Updating includes the following changes:

   - Change the base URL from `pubads.g.doubleclick.net` to `serverside.doubleclick.net`.
   - Add the parameter `ssss=mediatailor`. This indicates that MediaTailor is the server-side stitching source.
   - Remove the `IP` parameter. MediaTailor automatically passes in the end-user IP address using the `X-Forwarded-For` header.
   - Remove the `ss_req=1` parameter.

   For updated and complete VAST URL guidance, see the Server-side implementation guide or contact your Google account team.

Client-side AWS Elemental MediaTailor integration with Google Ad Manager

A MediaTailor client-side integration is required to use the Google Ad Manager Programmatic Access Libraries (PAL) SDKs. This integration is required if you want to use Ad Manager's open auction transaction type.

The PAL SDKs provide information about the content, device, and user data for a playback session. Through the PAL SDK, you can provide this information to Google Ad Manager, which can then make better determinations of what targeted ads to show. SDKs are available for Android, iOS, HTML5, and Cast. For information about using the PAL SDKs, see Google Ad Manager PAL SDK.

To create client-side integration with Ad Manager

1. Use the PAL SDK to generate a nonce.

The nonce is an encrypted string that PAL generates for stream requests. Each request must have a unique nonce. For information about setting up a nonce, choose your SDK from Google Ad Manager PAL SDK.

2. Use the `givn` parameter in your ADS request to pass through the nonce value. To do this, update your ADS URL to include `&givn=[player_params.givn]`. For instructions, see Enabling client-side tracking.

Datazoom player SDKs

MediaTailor has partnered with Datazoom to provide free player SDKs to ease integrations with SDKs such as those offered in the Ad Manager PAL. For information about the Datazoom and MediaTailor partnership, see Datazoom free player SDKs.

To access the Datazoom player SDKs, use the contact information on the Datazoom with AWS site.

Using a CDN to optimize ad personalization and content delivery

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

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

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

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

Topics

- Integrating a CDN
- How AWS Elemental MediaTailor handles BaseURLs for DASH
- CDN best practices with AWS Elemental MediaTailor

Integrating a CDN

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

Step 1: (CDN) create routing behaviors

In the CDN, create behaviors and rules that route playback requests to MediaTailor. Use the following rules for all segment requests (content, normal ad avails, and pre-roll ad avails):

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

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

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

- (Optional) Create one behavior that routes *ad segment* requests to the internal Amazon CloudFront distribution where AWS Elemental MediaTailor stores transcoded ads. Base this on a rule that includes a phrase to differentiate ad segment requests from content segment requests. This step is optional because AWS Elemental MediaTailor provides a default configuration.

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

Example Ad segment routing

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

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

Step 2: (AWS Elemental MediaTailor) create a configuration with CDN mapping

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

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

  For example, consider the following settings.

  - **Video content source** in the MediaTailor configuration is `http://origin.com/contentpath/`
  - **CDN content segment prefix** is `https://CDN_Hostname/`

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

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

Step 3: (CDN) set up CDN for manifest and reporting requests

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

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

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

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

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

Requests follow these formats:

- HLS master manifest format

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

  Example

```
https://777788889999.mediatailor.us-east-1.amazonaws.com/v1/master/
a1bc06b59e9a570b3b6b886a763d15814a86f0bb/Demo/assetId.m3u8
```

- HLS manifest format

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

Example

```
https://777788889999.mediatailor.us-east-1.amazonaws.com/v1/manifest/
a1bc06b59e9a570b3b6b886a763d15814a86f0bb/c240ea66-9b07-4770-8ef9-7d16d916b407/0.m3u8
```

- DASH manifest format

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

Example

```
https://777788889999.mediatailor.us-east-1.amazonaws.com/v1/dash/
a1bc06b59e9a570b3b6b886a763d15814a86f0bb/Demo/0.mpd
```

- Format for ad reporting request for server-side reporting

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

Example

```
https://777788889999.mediatailor.us-east-1.amazonaws.com/v1/segment/
Demo/240ea66-9b07-4770-8ef9-7d16d916b407/0/440384
```

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

Example Routing

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

How AWS Elemental MediaTailor handles BaseURLs for DASH

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

AWS Elemental MediaTailor behavior for content segments:

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

For ad segments, AWS Elemental MediaTailor does the following:

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

CDN best practices with AWS Elemental MediaTailor

We highly recommend that you use a content distribution network (CDN) to cache content and ad segments, but personalized manifest responses must *not* be cached or shared between viewers. Use the following settings for manifest traffic in your CDN to make the most of the service:

- Set all **time to live (TTL)** settings to **`0`**. This includes the maximum, minimum, and default TTL.
- **Forward all query strings** to MediaTailor. This way, all ad variables can be passed to the ad decision server (ADS) to determine the ads to be used in this playback session.

- **Forward the `User-Agent` header** to MediaTailor. The ADS often needs to know what user agent is requesting the content. If you don't forward the `User-Agent` header, the value that MediaTailor receives is the user agent of your CDN.

Customizing ad break behavior with ad suppression

When you create a configuration in AWS Elemental MediaTailor, you can specify optional ad break configuration settings that govern the behavior of ad breaks, including the ability to configure ad break suppression. This allows you to tailor the ad break experiences for your video content to meet your specific requirements.

Topics

- Configuring ad break suppression

Configuring ad break suppression

> **Note**
>
> Ad suppression is only available for live workflows.

You can configure MediaTailor to skip ad break personalization for live content. This is known as *ad suppression*, or *avail suppression*. This topic shows you how, and it also explains how configuring ad suppression works.

Ad suppression can be used for the following use cases:

- **Large manifest lookback window** – If a viewer starts playback at the live edge of a manifest but the lookback window is large, you might want to only insert ads starting after the viewer started watching. Or, insert ads for a portion of the total lookback window in the manifest. You can configure ad suppression so that MediaTailor personalizes ad breaks on or within a specified time range behind the live edge.
- **Mid-break join** – If the viewer starts watching a live video stream in the middle of an ad break, that user is likely to change the channel and not watch the advertisement. With ad suppression, you can skip ad break personalization if the ad break started before the viewer joined the stream.

Configuring ad suppression

To use ad suppression, you configure an **avail suppression mode**, **avail suppression value**, and **avail suppression fill policy** in the following ways:

- In the MediaTailor console
- Using the AWS Command Line Interface (AWS CLI)
- Using MediaTailor API, or as parameters in your client's playback session request

For information about configuration with parameters, see Configuring ad suppression parameters – playback session request.

Ad suppression configuration parameters

You can choose to turn on or turn off ad suppression. If you turn on ad suppression, you specify whether that suppression happens after the live playback edge or before the live playback edge of a live stream. In either case, you also specify a time, relative to the live edge, where MediaTailor doesn't personalize ads. When you turn on avail suppression, you can specify an avail suppression policy that MediaTailor uses for partial ad-break fills when a session starts mid-break.

The following are the ad suppression configuration parameters:

- **Avail suppression mode** – Sets the ad suppression mode. By default, ad suppression is off. **Accepted values**: `OFF`, `BEHIND_LIVE_EDGE`, or `AFTER_LIVE_EDGE`.
  - `OFF`: There is no ad suppression and MediaTailor personalizes all ad breaks.
  - `BEHIND_LIVE_EDGE`: MediaTailor doesn't personalize ad breaks that start before the live edge, minus the **Avail suppression value**.
  - `AFTER_LIVE_EDGE`: MediaTailor doesn't personalize ad breaks that are within the live edge, plus the **Avail suppression value**.
- **Avail suppression value** – A time relative to the live edge in a live stream. **Accepted value**: A time value in `HH:MM:SS`.
- **Avail suppression fill policy** – Defines the policy that MediaTailor applies to the **Avail suppression mode**. **Accepted values**: `PARTIAL_AVAIL`, `FULL_AVAIL_ONLY`.
  - `BEHIND_LIVE_EDGE` mode always uses the `FULL_AVAIL_ONLY` suppression policy.
  - `AFTER_LIVE_EDGE` mode can be used to invoke `PARTIAL_AVAIL` ad break fills when a session starts mid-break.

Ad suppression settings examples

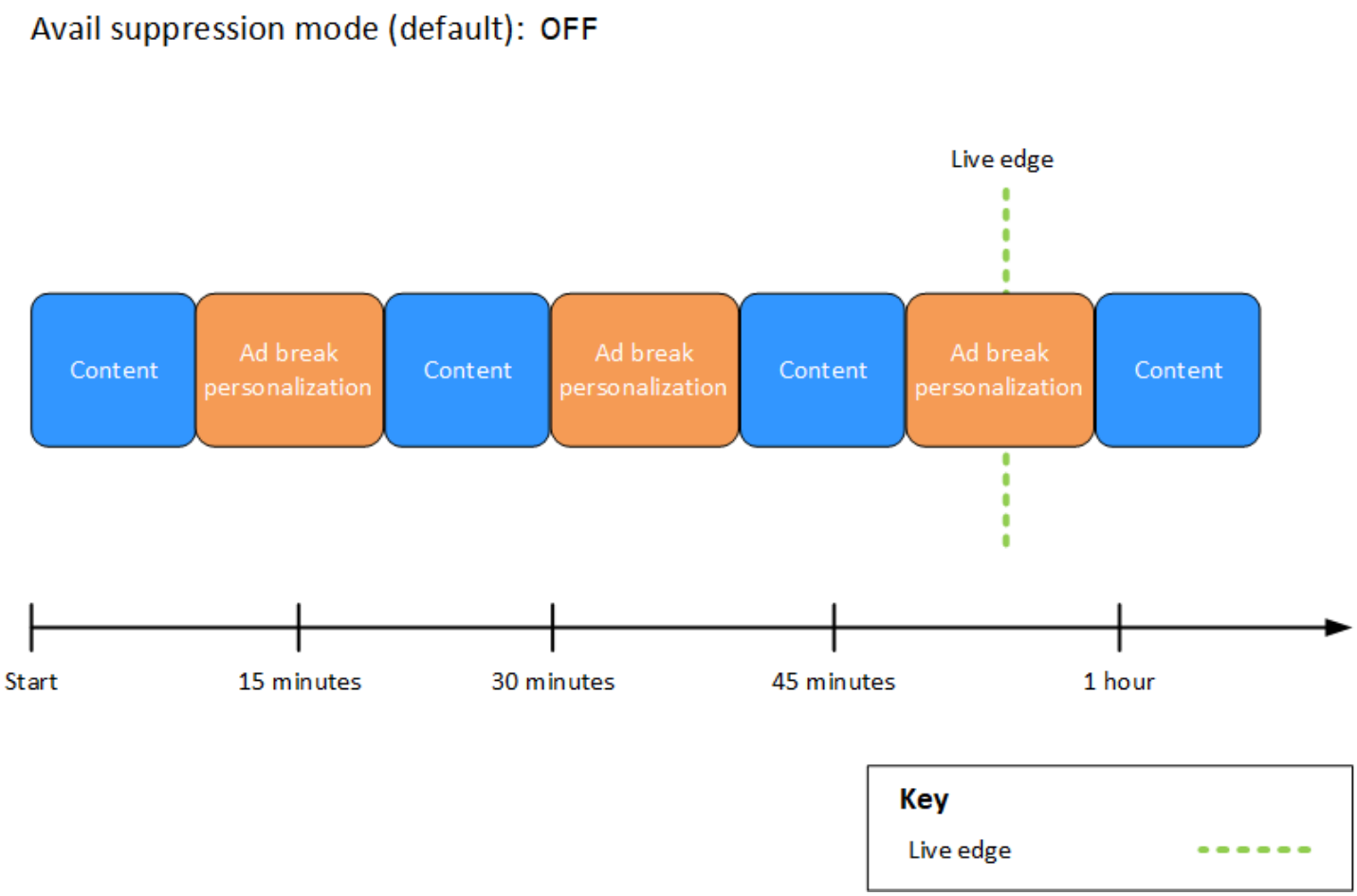
The way in which the ad suppression configuration parameters interact with one another lets you specify several different ways to handle ad suppression and avail filling before, at, or after the live edge of the live stream. This section provides examples that show you some of these interactions. Use these examples to help you set up the configuration parameters for your particular situation.

The following are examples of ad suppression settings:

Example 1: No ad suppression

When the **avail suppression mode** is OFF, there is no ad suppression and MediaTailor personalizes all ad breaks.

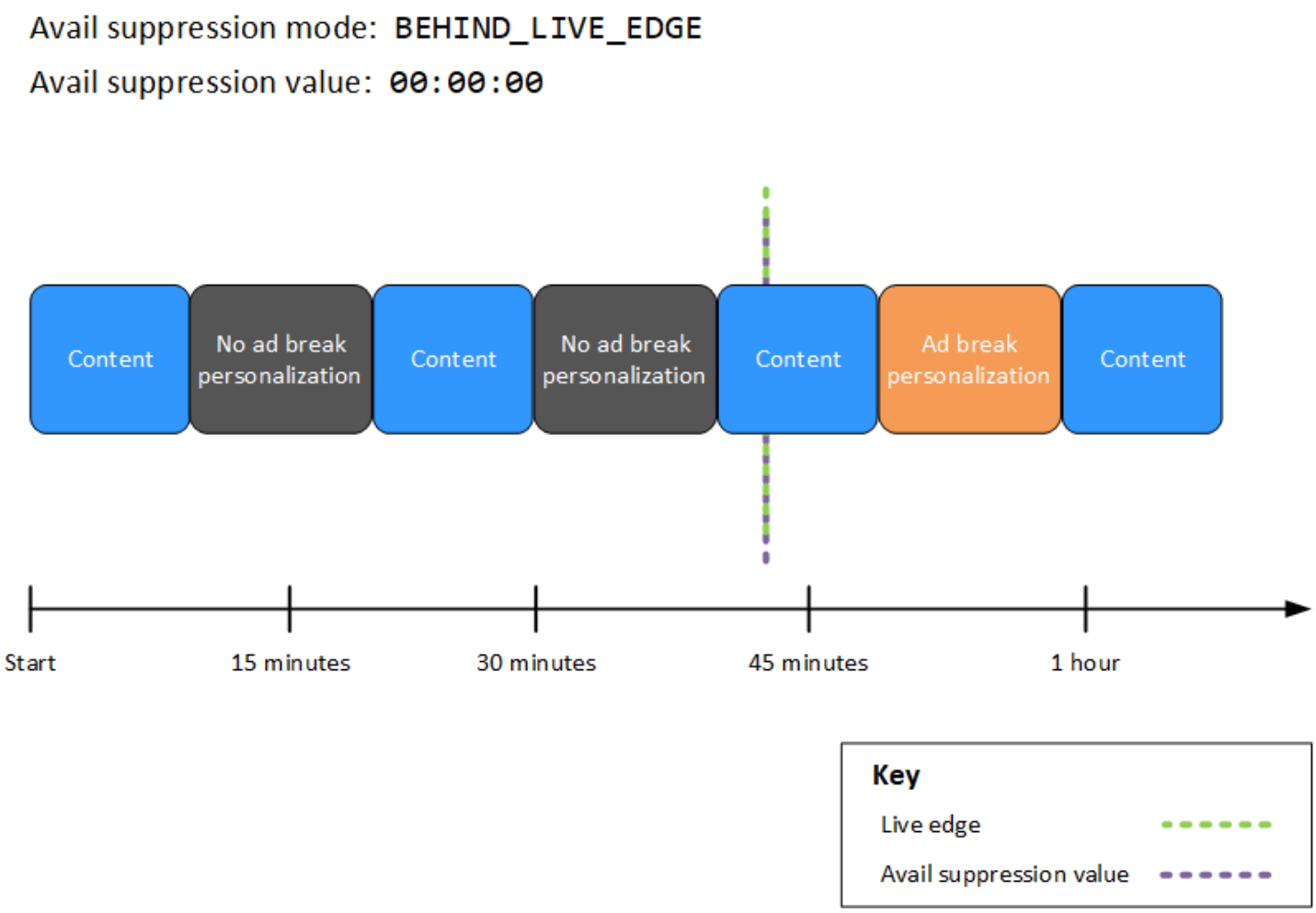
In the following figure, various blocks are arranged horizontally along a timeline that progresses from left to right. Each block represents a portion of time where the content of the live stream or a personalized ad break plays. A dotted line represents the current live edge of the live stream. Two ad breaks occur before the live edge, and another ad break is in progress at the live edge. As shown in the figure, when the avail suppression mode is OFF, MediaTailor personalizes all ad breaks that occur before the live edge on the timeline. MediaTailor also personalizes the ad break in progress at the live edge.



Example 2: `BEHIND_LIVE_EDGE` ad suppression with value in sync with live edge

When **avail suppression mode** is set to `BEHIND_LIVE_EDGE` and the **avail suppression value** is set to `00:00:00`, the avail suppression value is in sync with the live edge. MediaTailor doesn't personalize any ad breaks that start on or before the live edge.

In the following figure, various blocks are arranged horizontally along a timeline that progresses from left to right. Each block represents a portion of time where the content of the live stream, a personalized ad break, or a non-personalized ad break plays. A dotted line represents the current live edge of the live stream. Another dotted line, representing the avail suppression value set to `00:00:00`, overlaps the dotted line for the live edge. Two ad breaks occur before the live edge, and another ad break occurs after the live edge. As shown in the figure, when the avail suppression mode is set to `BEHIND_LIVE_EDGE`, and the avail suppression value is set to `00:00:00` so that it's in sync with the live edge, MediaTailor doesn't personalize any ad breaks that occur before the live edge on the timeline. MediaTailor personalizes the ad break that occurs *after* the live edge.



Example 3: `BEHIND_LIVE_EDGE` ad suppression with value behind live edge

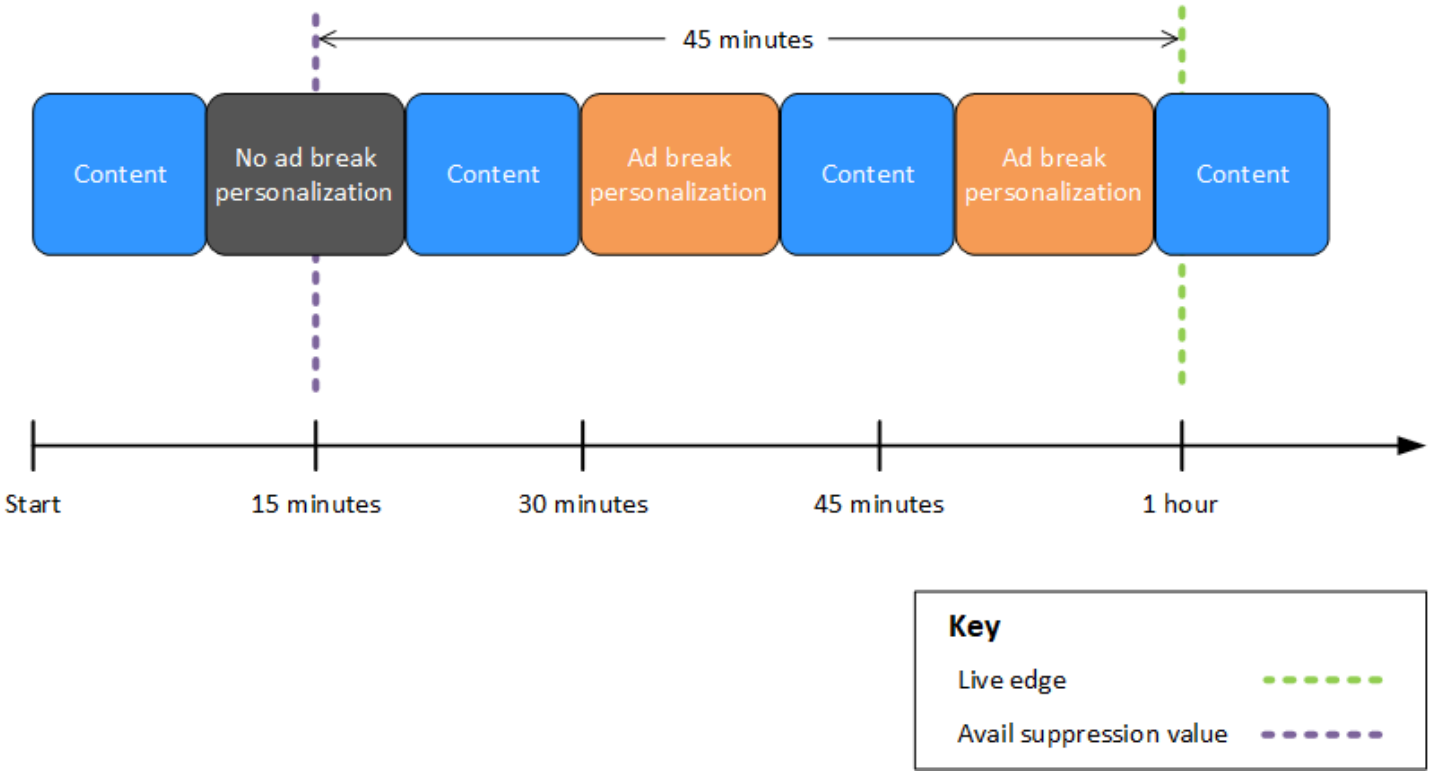
When the **avail suppression mode** is set to `BEHIND_LIVE_EDGE`, MediaTailor doesn't personalize any ad breaks on or before that time. In this example, MediaTailor personalizes ad breaks that start within 45 minutes behind the live edge. MediaTailor *doesn't* personalize ad breaks that start on or after 45 minutes behind the live edge.

In the following figure, various blocks are arranged horizontally along a timeline that progresses from left to right. Each block represents a portion of time where the content of the live stream, a personalized ad break, or a non-personalized ad break plays. A dotted line represents the current live edge of the live stream. Another dotted line, representing the avail suppression value set to `00:45:00`, occurs 45 minutes earlier in the timeline with respect to the dotted line for the live edge. The 45-minute time period between the dotted lines represents the avail suppression period. An ad break is in progress at the beginning of the avail suppression period. Two other ad breaks occur during the avail suppression period. As shown in the figure, when the avail suppression mode is set to `BEHIND_LIVE_EDGE`,

and the avail suppression value is set to `00:45:00` behind the live edge, MediaTailor personalizes any ad breaks that occur within the avail suppression period. MediaTailor *doesn't* personalize the ad break in progress at the beginning of the avail suppression period.

Avail suppression mode: **`BEHIND_LIVE_EDGE`**

Avail suppression value: **`00:45:00`**

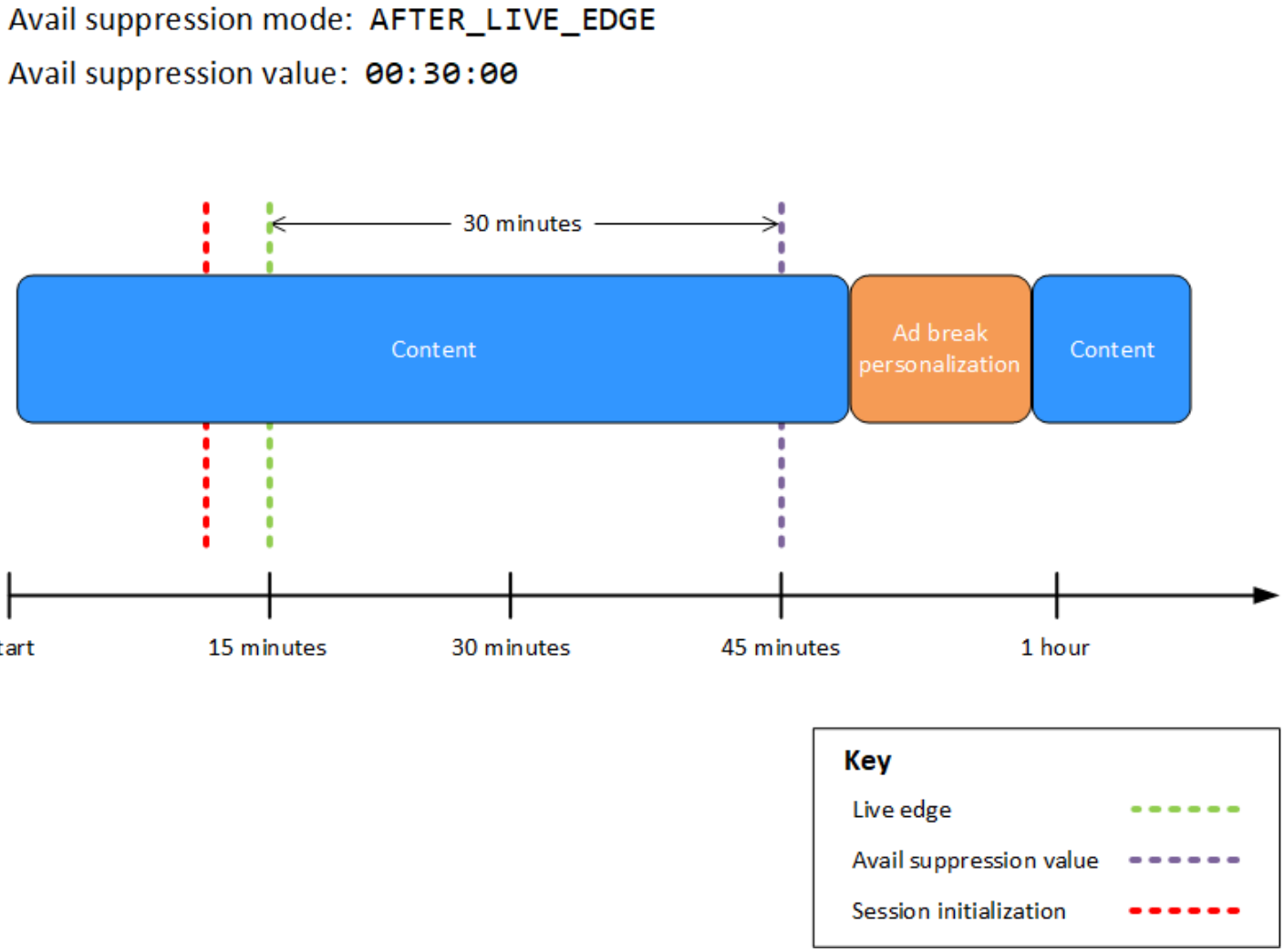


Example 4: `AFTER_LIVE_EDGE` ad suppression with no ad breaks occurring during the avail suppression period

When the **avail suppression mode** is set to `AFTER_LIVE_EDGE` and the **avail suppression value** is greater than zero, MediaTailor doesn't personalize any ad breaks until the elapsed time of the session has reached that value.

In the following figure, various blocks are arranged horizontally along a timeline that progresses from left to right. Each block represents a portion of time where the content of the live stream or a personalized ad break plays. A dotted line represents the current live edge of the live stream. Another dotted line, representing the avail suppression value set to `00:30:00`, occurs 30 minutes later in the timeline with respect to the dotted line for the live edge. A third dotted line, representing the session initialization, occurs earlier in the

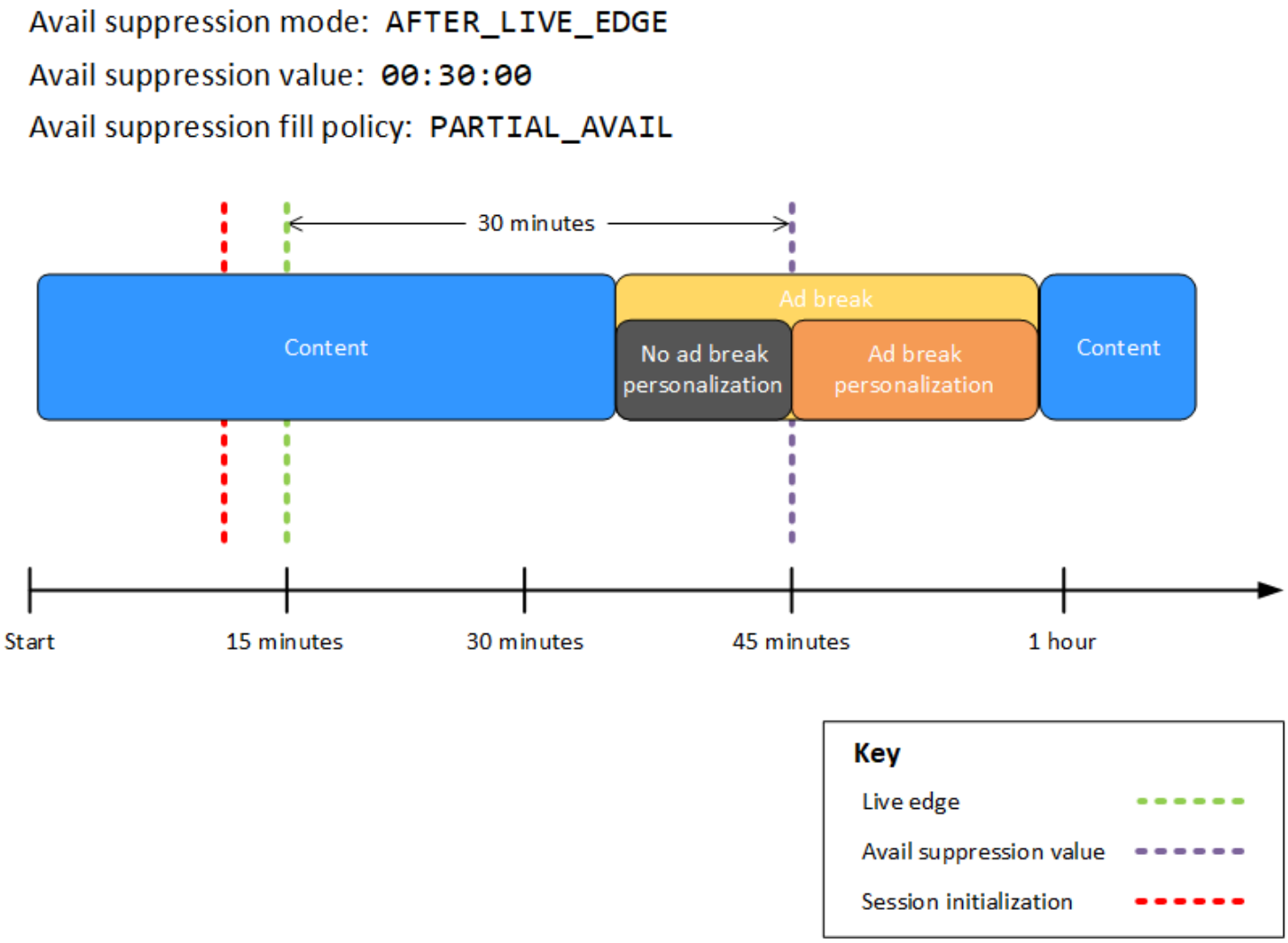
timeline with respect to the dotted line for the live edge. The 30-minute time period between the live-edge time and the avail-suppression-value time represents the avail suppression period. An ad break occurs after the avail suppression period. As shown in the figure, when the avail suppression mode is set to `AFTER_LIVE_EDGE`, the avail suppression value is set to `00:30:00` after the live edge, and the session initialization occurs before the live edge, MediaTailor personalizes any ad breaks that occur *after* the avail suppression period.



Example 5: `AFTER_LIVE_EDGE` ad suppression with `PARTIAL_AVAIL` fill policy and an ad break in progress at the end of the avail suppression period

When the **avail suppression mode** is set to `AFTER_LIVE_EDGE` and the **avail suppression value** is greater than zero, MediaTailor doesn't personalize any ad breaks until the elapsed time of the session has reached that value.

In the following figure, various blocks are arranged horizontally along a timeline that progresses from left to right. Each block represents a portion of time where the content of the live stream, a personalized ad break, or a non-personalized ad break plays. A dotted line represents the current live edge of the live stream. Another dotted line, representing the avail suppression value set to `00:30:00`, occurs 30 minutes later in the timeline with respect to the dotted line for the live edge. A third dotted line, representing the session initialization, occurs earlier in the timeline with respect to the dotted line for the live edge. The 30-minute time period between the live-edge time and the avail-suppression-value time represents the avail suppression period. An ad break is in progress at the end of the avail suppression period. As shown in the figure, when the avail suppression mode is set to `AFTER_LIVE_EDGE`, the avail suppression value is set to `00:30:00` after the live edge, the avail suppression fill policy is set to `PARTIAL_AVAIL`, and the session initialization occurs before the live edge, MediaTailor personalizes any ad breaks that occur *after* the avail suppression period. For the ad break in progress at the end of the avail suppression period, MediaTailor personalizes the portion of that ad break which occurs *after* the avail suppression period, but doesn't personalize the portion of that ad break which occurs *during* the avail suppression period.

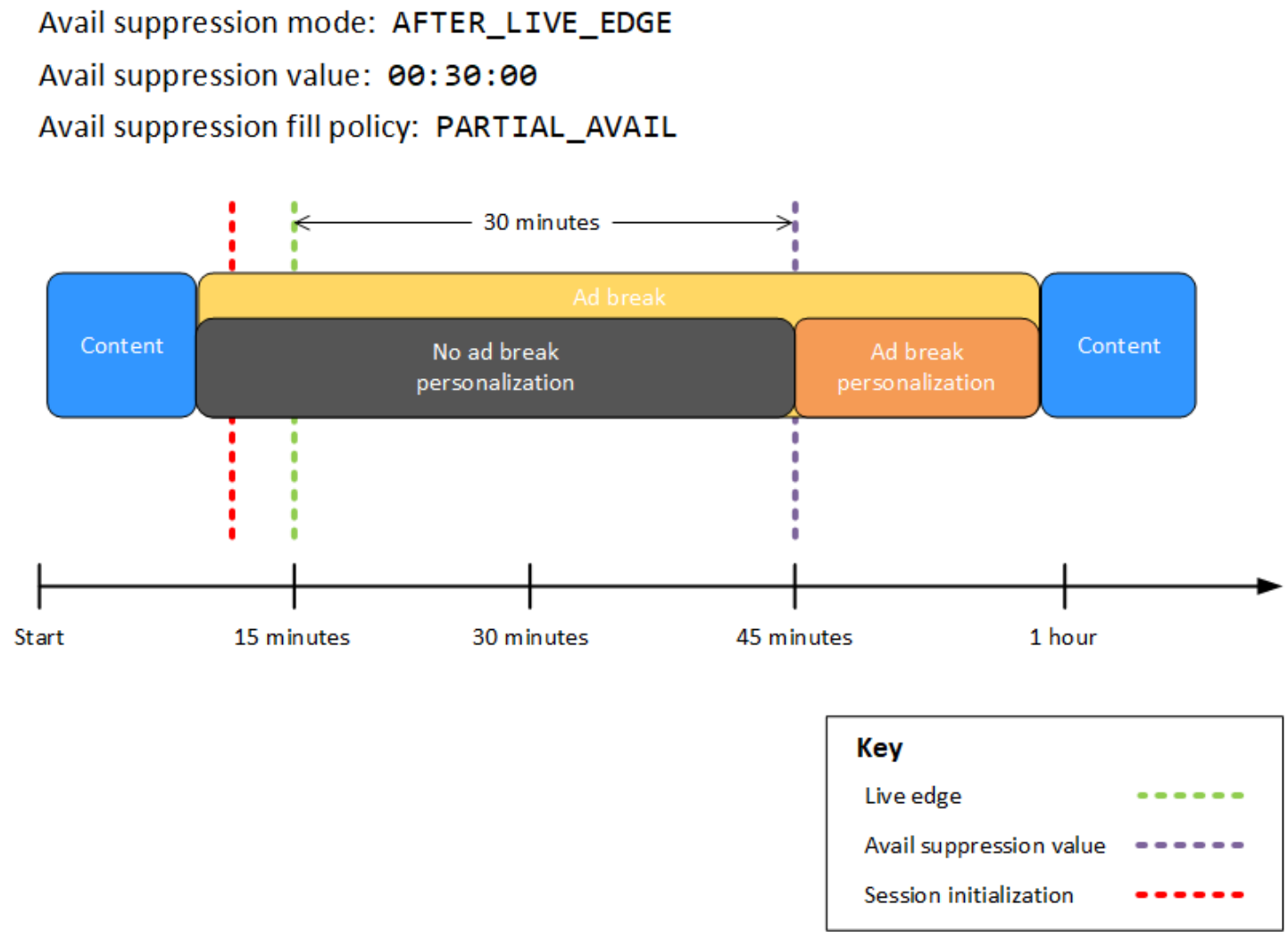


Example 6: `AFTER_LIVE_EDGE` ad suppression with `PARTIAL_AVAIL` fill policy and an ad break in progress from before session initialization to after the end of the avail suppression period

When the **avail suppression mode** is set to `AFTER_LIVE_EDGE` and the **avail suppression value** is greater than zero, MediaTailor doesn't personalize any ad breaks until the elapsed time of the session has reached that value.

In the following figure, various blocks are arranged horizontally along a timeline that progresses from left to right. Each block represents a portion of time where the content of the live stream, a personalized ad break, or a non-personalized ad break plays. A dotted line represents the current live edge of the live stream. Another dotted line, representing the avail suppression value set to `00:30:00`, occurs 30 minutes later in the timeline with respect to the dotted line for the live edge. A third dotted line, representing the session initialization, occurs earlier in the timeline with respect to the dotted line for the live edge. The 30-minute time period between the live-

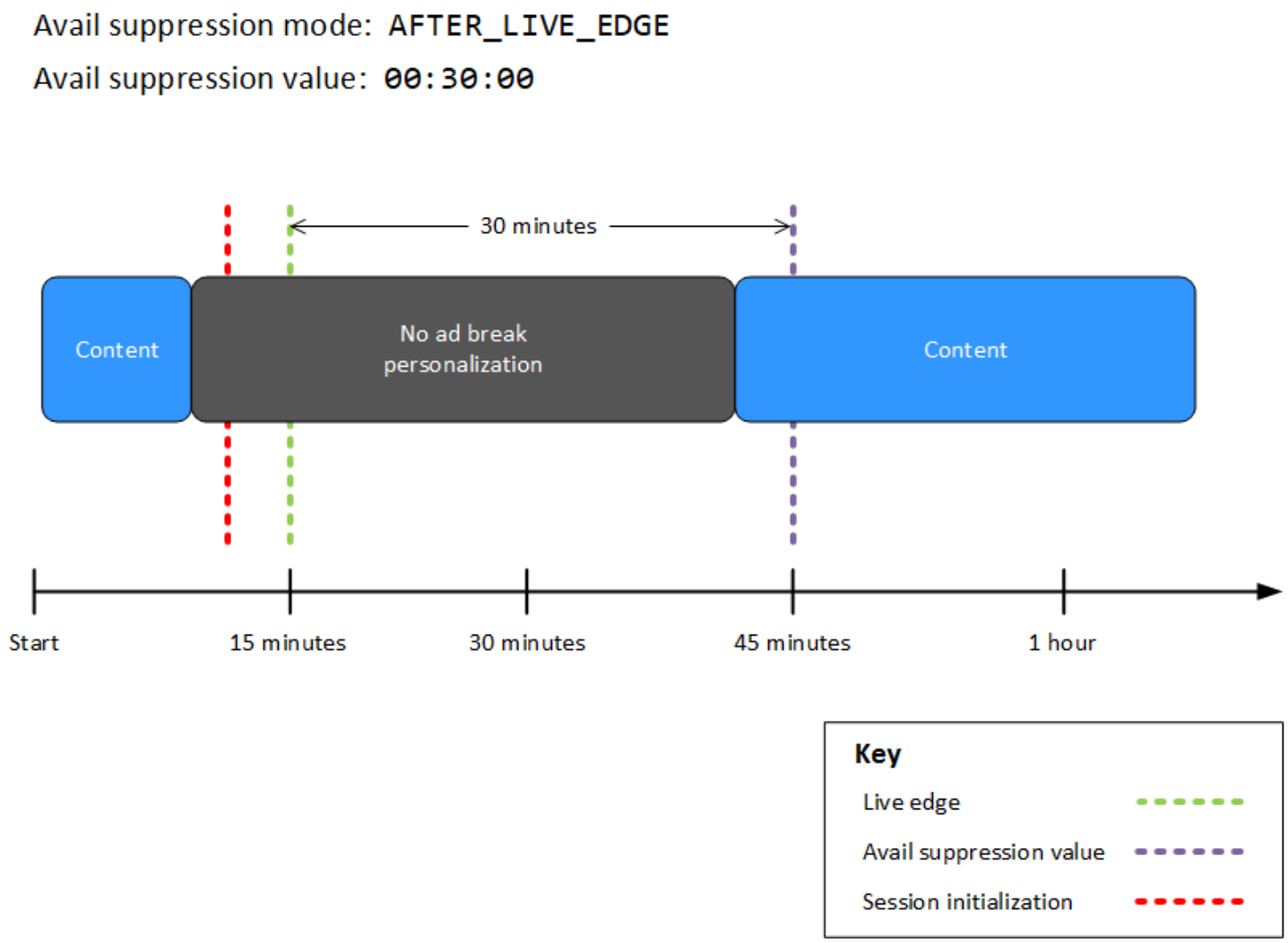
edge time and the avail-suppression-value time represents the avail suppression period. An ad break is in progress from a time before session initialization to a time after the avail suppression period. As shown in the figure, when the avail suppression mode is set to `AFTER_LIVE_EDGE`, the avail suppression value is set to `00:30:00` after the live edge, the avail suppression fill policy is set to `PARTIAL_AVAIL`, and the session initialization occurs before the live edge, MediaTailor personalizes any ad breaks that occur *after* the avail suppression period. For the ad break in progress before, during, and after the avail suppression period, MediaTailor personalizes the portion of that ad break which occurs *after* the avail suppression period, but doesn't personalize the portion of that ad break which occurs *before* or *during* the avail suppression period.



Example 7: `AFTER_LIVE_EDGE` ad suppression with an ad break in progress at the beginning of the avail suppression period

When the **avail suppression mode** is set to `AFTER_LIVE_EDGE` and the **avail suppression value** is greater than zero, MediaTailor doesn't personalize any ad breaks until the elapsed time of the session has reached that value.

In the following figure, various blocks are arranged horizontally along a timeline that progresses from left to right. Each block represents a portion of time where the content of the live stream or a non-personalized ad break plays. A dotted line represents the current live edge of the live stream. Another dotted line, representing the avail suppression value set to `00:30:00`, occurs 30 minutes later in the timeline with respect to the dotted line for the live edge. A third dotted line, representing the session initialization, occurs earlier in the timeline with respect to the dotted line for the live edge. The 30-minute time period between the live-edge time and the avail-suppression-value time represents the avail suppression period. An ad break is in progress from a time before session initialization to a time within the avail suppression period. As shown in the figure, when the avail suppression mode is set to `AFTER_LIVE_EDGE`, the avail suppression value is set to `00:30:00` after the live edge, and the session initialization occurs before the live-edge time but after the start of the ad break, MediaTailor doesn't personalize that ad break.



Configuring ad suppression parameters – playback session request

You can configure ad suppression settings via parameters in your *initial* server-side or client-side playback session request to MediaTailor. If you've already configured ad suppression settings via the MediaTailor Console or AWS Elemental MediaTailor API, these parameters override those settings.

Both the avail suppression mode and avail suppression value are required for ad suppression to work. These parameters can't be configured from different sources. For example, you can't configure one parameter with the MediaTailor console and another with a query parameter.

MediaTailor supports the following ad suppression parameters.

Name	Description	Accepted Values
availSuppressionMode	Sets the mode for ad suppression. By default, ad suppression is OFF. When set to BEHIND_LIVE_EDGE , MediaTailor doesn't fill ad breaks on or behind the aws.availSuppressionValue time. When set to AFTER_LIVE_EDGE , MediaTail or doesn't fill ad breaks on or behind the avail suppression period. The avail suppression period spans from the live-edge time to the aws.availSuppressionValue time, plus additional buffer time.	• OFF • BEHIND_LIVE_EDGE • AFTER_LIVE_EDGE
availSuppressionValue	A time relative to the live edge in a live stream.	A UTF-8 URL-encoded time code in HH:MM:SS. For example, 1 hour and 30 minutes would be 01%3A30%3A00 .
availSuppressionFillPolicy	Defines the policy to apply to the avail suppression mode. BEHIND_LIVE_EDGE always uses the full avail suppression policy. AFTER_LIVE_EDGE can be used to invoke partial ad break fills when a session starts mid-break.	• PARTIAL_AVAIL - not available for the BEFORE_LIVE_EDGE suppression mode • FULL_AVAIL_ONLY - the default value for the AFTER_LIVE_EDGE suppression mode

Server-side configuration

The base query parameter is `aws.availSuppression`, which is followed by optional parameter name and value pairs. To construct the query, append `aws.availSuppression=` to the end of the playback session request to MediaTailor, followed by parameter names and values. For more information about how to construct a server-side playback session request, see Server-side ad tracking.

Example: HLS

```
GET <mediatailorURL>/v1/master/<hashed-account-id>/<origin-id>/index.m3u8?
aws.availSuppressionMode=BEHIND_LIVE_EDGE&aws.availSuppressionValue=00%3A00%3A21
```

Server-side query syntax is listed in the following table.

Query String Component	Description
?	A restricted character that marks the beginning of a query.
`aws.`	The base query, which is followed by parameters constructed of name and value pairs. For a list of all of the available parameter s, see Configuring ad suppression parameters – playback session request.
=	Associates the parameter name with a value. For example, `aws.availSuppressionMode= BEHIND_LIVE_EDGE` .
&	Concatenates query parameters. For example, `aws.avail SuppressionMode= BEHIND_LIVE_EDGE &aws.avail SuppressionValue= 00:30:00&aws.availSuppress ionFillPolicy= FULL_AVAIL_ONLY` >.

Client-side configuration

Include `availSuppression` parameters in your client's POST request to MediaTailor. For more information about how to construct a client-side playback session request, see Client-side ad tracking.

Example: HLS

```
POST parent.m3u8
    {
       "availSuppression": {
          "mode": "BEHIND_LIVE_EDGE",
          "value": "00:00:21",
          "fillPolicy": "FULL_AVAIL_ONLY"
       }
    }
```

Inserting bumpers

Bumpers are short, non-skippable video or audio clips that play at the start or before the end of an ad break.

The following conditions apply to bumpers:

- Bumpers must be 10 seconds or less.
- Bumpers can be inserted at the start of an ad break, directly before the end of an ad break, or both.
- Bumpers play during every ad break in a playback session unless pre-roll is configured. If pre-roll is configured, bumpers won't play during the pre-roll break. Instead, they will play in every subsequent break after the pre-roll.
- For HLS, you must include the `duration` attribute with each SCTE-35 `EXT-X-CUE-OUT` tag.
- Bumpers are transcoded to match source content.
- You are not charged for bumpers.

Configuring bumpers

To use bumpers, configure the bumper URLs with the MediaTailor console, MediaTailor API, or the AWS Command Line Interface (AWS CLI). You can configure a start bumper, an end bumper, or both. Bumpers are stored on a server, such as Amazon Simple Storage Service (Amazon S3). The bumper URLs indicate the location of the stored bumper asset(s).

Example start and end bumper URLs:

Start bumper URL: `https://s3.amazonaws.com/startbumperad`

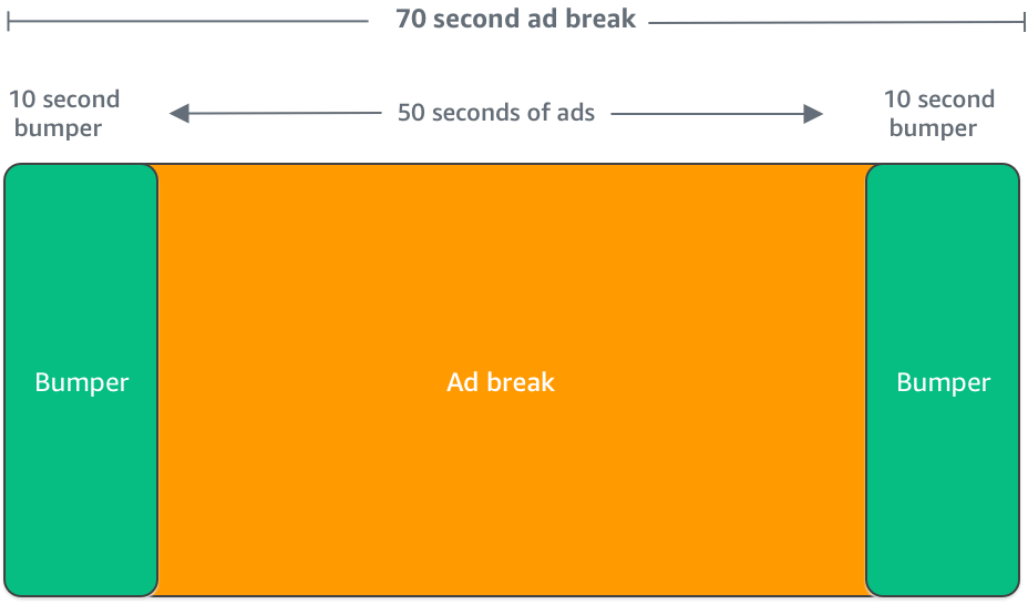
End bumper URL: `https://s3.amazonaws.com/endbumperad`

Example

The following is an example of bumper ad behavior.

Example 1: Start and end bumpers

In this example, start and end bumpers are enabled. The ad decision server has 50 seconds of personalized ads to fill a 70-second ad break. The 10-second start bumper plays at the start of the ad break, 50 seconds of ads plays, then the 10-second end bumper.



Inserting pre-roll ads

> **Note**
>
> Configurable pre-roll ads are only available for live workflows. For details about how ad insertion (including pre roll) works for VOD, see Ad stitching behavior for VOD.

MediaTailor can insert ads at the beginning of a playback session, before the main content begins. These are *pre-roll* ads.

To insert pre-roll ads, complete the **Live pre-roll ad decision server** and **Live pre-roll maximum allowed duration** fields in the **Additional** settings on your configuration, as described in Optional configuration settings.

1. When MediaTailor receives a playback request, it sends a request to for pre-roll ads based on the following fields in the MediaTailor playback configuration:
   - **Live pre-roll ad decision server** is the ad decision server (ADS) URL where MediaTailor sends the request for pre-roll ads.
   - **Live pre-roll maximum allowed duration** is the total maximum length of time for the pre-roll ads. MediaTailor takes the following action based on the maximum allowed duration:
     - If the total duration of the ads in the ADS response is *less* than the value you gave in **Live pre-roll maximum allowed duration**, MediaTailor inserts all of the ads. When the last ad is complete, MediaTailor immediately returns to the underlying content.
     - If the total duration of the ads in the ADS response is *more* than the value you gave in **Live pre-roll maximum allowed duration**, MediaTailor selects a set of ads that fit into the duration without going over. MediaTailor inserts these ads without clipping or truncation. MediaTailor returns to the underlying content when the last selected ad completes.
2. When MediaTailor receives the pre-roll response from the ADS, it manipulates the manifest to add links to the pre-roll ads. MediaTailor calculates the start time of the pre-roll ad break as follows:
   - For DASH, the formula is `(publishTime - availabilityStartTime) - max(suggestedPresentationDelay, minBufferTime)`.
   - For HLS, the formula is `max(2*EXT-X-TARGETDURATION, EXT-X-START:TIMEOFFSET)`.
3. MediaTailor determines what action to take on any ad breaks that aren't pre-rolls. If the pre-roll overlaps another ad break, MediaTailor doesn't personalize the overlapped portion of the ad break.

Inserting slate

> **Note**
>
> Slate is only available for live workflows.

With AWS Elemental MediaTailor, you can designate a *slate ad* for ad breaks. A slate is a default MP4 asset that is inserted into a stream, such as a still image or looped video, that plays instead of the live content.

AWS Elemental MediaTailor shows a slate in the following situations:

- To fill in time that's not fully used by an ad replacement
- If the ad decision server (ADS) responds with a empty VAST or VMAP response
- For error conditions, such as ADS timeout
- If the duration of ads is longer than the ad break
- If an ad isn't available

Configuring the slate

You designate the slate in the additional configuration pane in the MediaTailor console. MediaTailor downloads the slate from the URL that you specify, and transcodes it to the same renditions as your content. You can control the maximum amount of time that a slate will be shown through the optional **personalization threshold** configuration in the MediaTailor console. For more information, see the section called “Optional configuration settings”.

VPAID requirements

Configuring a slate is required if you use VPAID. For VPAID, MediaTailor inserts the slate for the duration of the VPAID ad. This duration might be slightly higher than the duration of the VPAID ad as reported by VAST to accommodate user interactivity. The video player then handles the VPAID ad based on the client-side reporting metadata that MediaTailor returns. For information about client-side reporting, see the section called “Client-side tracking”. For information about VPAID, see the section called “VPAID requirements”.

If you aren't using VPAID, if you don't configure a slate then MediaTailor defaults to the underlying content stream.

Prefetching ads

With ad prefetching, AWS Elemental MediaTailor proactively fetches ads from the ad decision server (ADS) and prepares them for upcoming ad breaks based on a pre-defined schedule. During live streams, ad request and transcoding timeouts can lead to decreased ad fill rates and missed

monetization opportunities. With ad prefetching, MediaTailor works with the ADS to determine what ads to fill breaks, before they happen. This increased time between ad decisioning and ad breaks provides more time for programmatic ad trading, and reduces ad insertion latency because both ad transcoding and ADS communications run in the background.

To set up ad prefetching, you create one or more *prefetch schedules* on your playback configuration. A prefetch schedule tells MediaTailor how and when to retrieve and prepare ads for an upcoming ad break. Each prefetch schedule defines a single set of ads for MediaTailor to place in a single ad break. To prefetch ads for multiple ad breaks, you can create multiple prefetch schedules. When you create a prefetch schedule, you can include criteria that gives you granular control over which ad break and which playback stream MediaTailor places the prefetched ads in.

The following topics describe more about ad prefetching.

For information about ad preconditioning (transcoding ads before they are needed), see Using preconditioned ads with AWS Elemental MediaTailor.

Topics

- How prefetching works
- Creating prefetch schedules
- Deleting prefetch schedules

How prefetching works

When your client makes a manifest request to MediaTailor, the service evaluates all of the prefetch schedules that are associated with the playback configuration. If MediaTailor doesn't find a matching prefetch schedule, the service reverts to normal ad insertion and doesn't prefetch ads.

If MediaTailor finds a matching prefetch schedule, the service evaluates the schedule based on two components, retrieval and consumption.

Retrieval

This defines the *retrieval window*, which is the time range when MediaTailor prefetches ads from the ADS. To set up the retrieval window, first determine when the ad break will occur.

For advanced use cases, you can optionally add *dynamic variables* to the prefetch request that MediaTailor sends to the ADS. This lets you send session, player, and other data to the ADS as part of the request. If you don't include dynamic variables in the prefetch schedule, MediaTailor

uses the dynamic variables, if any, that you configured in your playback configuration's ADS URL.

Consumption

This defines the *consumption window*, which is the time range when MediaTailor places prefetched ads into the ad break.

For this component, you can optionally add as many as five dynamic session variables for *avail matching criteria* to a prefetch schedule. MediaTailor uses these criteria to determine whether the ad break is eligible for placement of the prefetched ads. For example, you can use the *scte.event_id* dynamic variable if you want the service to place ads in an ad break with a specific SCTE event ID. MediaTailor places the prefetched ads into an ad break only if the ad break meets the criteria defined by the dynamic session variables.

For a list of supported avail-matching criteria, see the *Available for ad prefetch* column in the table on Using session variables.

When your client sends manifest requests to MediaTailor during the retrieval window, MediaTailor proactively sends requests to the ADS to retrieve and prepare the ads for later insertion. If you set up dynamic variables for retrieval, MediaTailor includes those variables in the requests.

When MediaTailor encounters an SCTE-35 ad break marker during the consumption window, the service uses the avail matching criteria, if configured, to determine which ad break to place the ads in. If avail matching criteria aren't configured, MediaTailor places the prefetched ads in the first ad break within the consumption window.

Understanding prefetching costs

For prefetch ad retrieval, you'll be charged at the standard transcoding rate for the prefetched ads that MediaTailor transcodes. For prefetch ad consumption, you'll be charged at the standard rate for ad insertion for the prefetched ads that MediaTailor places in ad breaks. For information about transcoding and ad insertion costs, see AWS Elemental MediaTailor Pricing.

Creating prefetch schedules

The following procedure explains how to create a prefetch schedule by using the MediaTailor console. For information about creating and managing prefetch schedules programmatically using the MediaTailor API, see PrefetchSchedules in the *AWS Elemental MediaTailor API Reference*.

> **Note**
>
> If you want to use avail matching criteria in a schedule, make sure that you first configure your playback configuration's ADS URL template with dynamic session variables, otherwise the avail matching criteria won't have an effect. For information about working with dynamic variables, see Step 3: Configure ADS request URL and query parameters in the *Getting started with MediaTailor* ad insertion topic.

To create a new prefetch schedule using the console

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Configurations**. Select the playback configuration that you want to create a prefetch schedule for.
3. On the **Prefetch schedules** tab, choose **Add prefetch schedule**.
4. Under the **Prefetch schedule details** pane, do the following:

   - For **Name**, enter an identifier for your prefetch schedule, such as **my-prefetch-schedule**.
   - For **Stream ID**, optionally enter a unique ID. If your origin contains multiple playback streams, you can use this ID to instruct MediaTailor to place ads in a specific stream. For example, if your origin has a sports stream and a TV show stream, you can use the stream ID to create prefetch schedules to insert ads targeted for the sports stream. You pass the stream ID value to MediaTailor in your client's session initialization or manifest request. For more information see the following example.
     - For *server-side tracking*, include the `?aws.streamId` query parameter and value in your client's `GET` HTTP request to your MediaTailor endpoint. For general information about server-side tracking see Server-side ad tracking. A manifest request to an HLS endpoint that includes a stream ID looks like the following, where *`myStreamId`* is the name of your stream ID:

       ```
       GET <mediatailorURL>/v1/master/<hashed-account-id>/<origin-id>/<asset-id>?
       aws.streamId=myStreamId
       ```

     - For *client-side tracking*, include the `streamId` key and value in your client's `POST` HTTP session initialization request body to the **MediaTailor/v1/session** endpoint. For general information about client-side tracking see Client-side ad tracking. A session initialization

request that includes a stream ID looks like the following, where *myStreamId* is the name of your stream ID:

```
POST <mediatailorURL>/v1/session/<hashed-account-id>/<origin-id>/<asset-id>
{
    'streamId': 'myStreamId'
}
```

5. On the **Retrieval** pane, specify the retrieval settings you want to use. These settings determine when MediaTailor prefetches ads from the ADS. They also determine which dynamic session variables to include in the request to the ADS, if any.

   - For **Start time**, enter the time when MediaTailor can start prefetch retrievals for this ad break. MediaTailor will attempt to prefetch ads for manifest requests made by your client on or after this time. The default value is the current time. If you don't specify a value, the service starts prefetch retrieval as soon as possible.
   - For **End time**, enter the time when you want MediaTailor to stop prefetching ads for this ad break. MediaTailor will attempt to prefetch ads for manifest requests that occur at or before this time. The retrieval window can overlap with the consumption window.
   - In the **Dynamic variables** section, enter as many as 100 dynamic session variables. MediaTailor uses these variables for substitution in prefetch requests that it sends to the ADS. If you don't enter any dynamic session variables, MediaTailor makes a best effort attempt to interpolate the values for the dynamic variables contained in your ADS URL.
     - Select **Add dynamic variable**.
     - For **Key**, enter a dynamic session variable key, such as `scte.event_id`. You can use any dynamic variable that MediaTailor supports. For information about dynamic session variables, see Using session variables.
     - For **Value**, enter a dynamic variable value, such as *my-event*.
     - To add another dynamic variable, choose Select **Add dynamic variable**.
6. On the **Consumption** pane, specify the settings that you want to use for the consumption window. These settings determine when MediaTailor places the ads into the ad break. They also determine any avail matching criteria that you want to use.

   - For **Start time**, enter the time when you want MediaTailor to begin to place prefetched ads into the ad break. the default value is the current time. If you don't specify a time, the service starts prefetch consumption as soon as possible.

- For **End time**, enter a time when you want MediaTailor to stop placing the prefetched ads into the ad break. MediaTailor will attempt to prefetch ads for your client's manifest requests that occur at or before this time. The end time must be after the start time, and less than one day from now. The consumption window can overlap with the retrieval window.
- In the **Avail matching criteria** section, select **Add avail criteria** and add as many ad five avail matching criteria to your schedule. Then, under **Dynamic variable key**, add a dynamic variable key, such as `scte.event_id`. MediaTailor will place the prefetched ads into the ad break only if it meets the criteria defined by the dynamic variable values that either your client passes to MediaTailor, or that MediaTailor infers from information such as session data. For information, see the preceding section avail-matching-criteria.

7. Select **Add avail criteria**.

Prefetch schedules automatically expire after the consumption window's end time. For diagnostic purposes, they remain visible for at least 7 days, after which MediaTailor automatically deletes them. Alternatively, you can manually delete a prefetch schedule at any time. For information about how to manually delete a prefetch schedule, see the following the section called “Deleting prefetch schedules” section.

Determining how often your client should call the CreatePrefetchSchedule API

Your client can programmatically call the CreatePrefetchSchedule API once per day to set up retrieval and consumption if you have knowledge of exactly when ad breaks will occur. Or, your client can call the API many times over the course of the day to define retrieval and consumption. When choosing an API call frequency, take into consideration the maximum number of active prefetch schedules, and the likelihood of whether your ad break schedule will change after you create your prefetch schedule(s). If it's likely that the ad break schedule will change after you've created your prefetch schedule(s), you might want to call the API more frequently.

Deleting prefetch schedules

The following procedure explains how to delete a prefetch schedule by using the MediaTailor console. For information about how to delete prefetch schedules programmatically using the MediaTailor API, see DeletePrefetchSchedule in the *AWS Elemental MediaTailor API Reference*.

> **Note**
>
> Deletion doesn't occur in real time. You might experience a delay while MediaTailor deletes the prefetch schedule(s), during which time prefetch retrieval and consumption will continue to run in the background.

To delete a prefetch schedule using the console

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Configurations**. Select the playback configuration that contains the prefetch schedule(s) that you want to delete.
3. On the **Prefetch schedules** tab, select the prefetch schedule that you want to delete. Then, choose **Delete**.

Using preconditioned ads with AWS Elemental MediaTailor

In a typical ad insertion workflow, MediaTailor dynamically transcodes ads to match the content stream, saves them, and stitches the ads into the live stream. Because this process happens only after MediaTailor receives the ad in a VAST response from the ad decision server (ADS), there is a delay for when the ad is available for stitching. If additional latency is introduced to the ad stitching workflow (either because of ADS timeout or other content or network issues), MediaTailor could partially fill the avail, or miss the ad break altogether.

To reduce the time needed to stitch ads into your content, you can use preconditioned ads. A preconditioned ad is one that you transcode before using it in MediaTailor ad insertion. Instead of providing URLs for the unconditioned ads to your ADS, you provide the URLs for the preconditioned ads. In its VAST response to the MediaTailor request, the ADS includes direct links to the preconditioned ads. By removing the transcoding part of ad stitching, MediaTailor needs to just save the ad and stitch it into the content stream. The ad stitching process with preconditioned ads reduces the time between when MediaTailor is made aware of an ad through the VAST response and when the ad is stitched into the content.

Alternatively, you can also use ad prefetching, which is when you configure MediaTailor to perform the ad stitching process at a scheduled time before the ad break is needed. For more information about ad prefetching, see Prefetching ads.

Preconditioned ads requirements

The following are requirements to consider when setting up an ad stitching workflow with preconditioned ads.

`MediaFiles` requirements

The VAST response that the ad server sends to MediaTailor must include `MediaFiles` that meet these requirements:

The ad (`Creative`) must have variants that conform to the bitrate variants of the content stream.*It's your responsibility to ensure that the VAST response uses the right ad variants to match template manifests.*

While using preconditioned ads can help make ad insertion more efficient, MediaTailor does not have the ability to manage the transcoding process to ensure that the media files for the ads are compatible with the content manifest specifications. If the ad doesn't match the content stream, MediaTailor could miss the insertion or the mismatch could cause the playback device to error.

Additionally, to be stitched into the content stream without MediaTailor transcoding, a `MediaFile` must meet these requirements:

- It must be accessible on the public internet so that MediaTailor can download it.
- It must use streaming delivery, denoted as `delivery="streaming"` in the VAST response.
- It must be either a `.m3u8` (for HLS) or `.mpd` (for DASH) file.

Example VAST response

From the following example VAST response, MediaTailor inserts the `MediaFile` with the following URLs:

- For an HLS stream, MediaTailor uses `https://example-ad-origin.amazonaws.com/ad1/index_low.m3u8`. This is the first `MediaFile` with streaming delivery and a supported file extension (`.m3u8`).
- For a DASH stream, MediaTailor uses `https://example-ad-origin.amazonaws.com/ad1/index.mpd`. This is the first `MediaFile` with streaming delivery and a supported file extension (`.mpd`).

```
<?xml version="1.0" encoding="UTF-8"?>
<VAST xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="3.0">
    <Ad id="ad1">
        <InLine>
            <AdSystem>ExampleAdSystem</AdSystem>
            <AdTitle>ad1</AdTitle>
            <Impression><![CDATA[https://example-impression.amazonaws.com]]></
Impression>
            <AdServingId>de8e0d33-9c72-4d77-bb3a-f7e566ffc605</AdServingId>
            <Creatives>
                <Creative id="creativeId1" sequence="1">
                    <Linear skipoffset="00:00:05">
                        <Duration>00:00:30</Duration>
                        <MediaFiles>
                            <MediaFile delivery="progressive" width="1280" height="720"
 type="video/mp4" bitrate="533" scalable="true" maintainAspectRatio="true"><!
[CDATA[https://example-ad-origin.amazonaws.com/ad1/ad1.mp4]]></MediaFile>
                            <MediaFile delivery="streaming" width="1280"
 height="720" type="application/dash+xml" bitrate="533" scalable="true"
 maintainAspectRatio="true"><![CDATA[https://example-ad-origin.amazonaws.com/ad1/
index.mpd]]></MediaFile>
                            <MediaFile delivery="streaming" width="640"
 height="360" type="application/x-mpegURL" bitrate="262" scalable="true"
 maintainAspectRatio="true"><![CDATA[https://example-ad-origin.amazonaws.com/ad1/
index_low.m3u8]]></MediaFile>
                            <MediaFile delivery="streaming" width="2560"
 height="1440" type="application/x-mpegURL" bitrate="1066" scalable="true"
 maintainAspectRatio="true"><![CDATA[https://example-ad-origin.amazonaws.com/ad1/
index_high.m3u8]]></MediaFile>
                        </MediaFiles>
                    </Linear>
                </Creative>
            </Creatives>
        </InLine>
    </Ad>
</VAST>
```

Ad manifest requirements

To use preconditioned ads, your parent and child ad manifests must meet these requirements:

- The manifest that's linked in the `Creative` section of the VAST response must be the parent ad manifest.

- The URLs for the child ad manifests must be relative paths.
- The child ad manifests must be in the same directory as the parent manifest, at the same level. Child manifests can't be in a sub-directory or other location.

Example supported parent manifest

The following parent ad manifest contains relative URLs for child ad manifests. The child manifests are also in the same directory as the parent manifest.

```
#EXTM3U
#EXT-X-STREAM-INF:BANDWIDTH=150000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
index_1.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=440000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
index_2.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=640000,RESOLUTION=640x360,CODECS="avc1.42e00a,mp4a.40.2"
index_3.m3u8
```

Example unsupported parent manifest: subdirectories

The following parent ad manifest contains child manifests that are in sub-directories relative to the parent manifest. It is not a supported manifest for preconditioned ads.

```
#EXTM3U
#EXT-X-STREAM-INF:BANDWIDTH=150000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
child/index_1.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=440000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
child/index_2.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=640000,RESOLUTION=640x360,CODECS="avc1.42e00a,mp4a.40.2"
child/index_3.m3u8
```

Example unsupported parent manifest: absolute URLs

The following parent ad manifest contains child manifests with absolute URLs. It is not a supported manifest for preconditioned ads.

```
#EXTM3U
#EXT-X-STREAM-INF:BANDWIDTH=150000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
https://example.mediatailor.us-west-2.amazonaws.com/index_1.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=440000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
https://example.mediatailor.us-west-2.amazonaws.com/index_2.m3u8
```

```
#EXT-X-STREAM-INF:BANDWIDTH=640000,RESOLUTION=640x360,CODECS="avc1.42e00a,mp4a.40.2"
https://example.mediatailor.us-west-2.amazonaws.com/index_3.m3u8
```

Preconditioned ads workflow

The following is a basic description of how preconditioned ads work in an ad stitching workflow with MediaTailor. The first part of the workflow are actions that you must take to get set up for using preconditioned ads. The second part describes how MediaTailor processes the ads.

Part 1: Preconditioned ads set up

Complete the following steps to set up a workflow that uses preconditioned ads in MediaTailor.

1. Use a transcoder service, such as AWS Elemental MediaConvert, to condition your creatives into variants that support the different bitrates, resolutions, and codecs of your template manifests.
2. Provide the URLs for the pre-transcoded media files to your ADS, for use in VAST responses.
3. Create your playback configuration in MediaTailor. To use preconditioned ads, select **None** for the **Streaming media file conditioning** setting in the configuration.
4. Continue with your content delivery set up as you normally would.

Part 2: MediaTailor ad processing

MediaTailor ad stitching completes as described in How MediaTailor ad insertion works. When MediaTailor receives a VAST response from the ADS, it uses the following logic to determine what actions to take for the ads. This logic is dictated by the **Streaming media file conditioning** setting on the playback configuration.

- When **Streaming media file conditioning** is set to **Transcode**, MediaTailor transcodes the media files with `progressive` delivery and stitches them into the manifest. If there aren't enough ads with `progressive` delivery media files to fill the avail, MediaTailor transcodes and uses those with `streaming` delivery.
- When **Streaming media file conditioning** is set to **None**, MediaTailor stitches ads with `streaming` delivery media files into the manifest without transcoding them. If there aren't enough ads with `streaming` delivery media files to fill the avail, MediaTailor transcodes and uses those with `progressive` delivery.

Using dynamic ad variables in MediaTailor

The AWS Elemental MediaTailor request to the ad decision server (ADS) includes information about the current viewing session, which helps the ADS choose the best ads to provide in its response. When you configure the ADS template in your MediaTailor configuration, you can include dynamic variables, also known as macros. Dynamic variables are replaceable strings.

Dynamic variables can 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.
- **Domain variables** – Dynamic variables that can be used for URL domains, such as the **my-ads-server.com** part of the URL http://my-ads-server.com. For details, see Using domain variables.
- **Session data** – Dynamic values that are provided by MediaTailor for each session, for example, the session ID. For details, see Using session variables.
- **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 Using player variables.

Passing parameters to the ADS

The following steps describe how to set up dynamic variables in MediaTailor requests to the ADS.

- For information about supported formatting for query parameters, see Manifest query parameter supported characters and limitations and ADS query parameter length limitations.
- For additional customizations to the ADS request, see Advanced usage.

To pass session and player information to the ADS

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

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

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

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

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

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

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

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

     HLS example:

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

     DASH example:

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

   - (Option) Call for client-side ad-tracking reporting – Provide parameters for the ADS inside an `adsParams` object.

     HLS example:

     ```
     POST master.m3u8
         {
             "adsParams": {
     ```

```
            "deviceType": "ipad"
        }
    }
```

DASH example:

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

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

Example MediaTailor requests with ad variables

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

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

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

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

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

  - DASH example:

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

Manifest query parameter supported characters and limitations

You can use the following characters in query parameters that are used in manifest requests:

- Alphanumeric (A-Z, a-z, 0-9)
- Periods (.)
- Hyphens (-)
- Underscores (_)
- Back slashes (\)

Length limitations

The total length of all manifest query parameters (the key and value combined) must not exceed 2000 characters.

Unsupported characters

You can't use the following characters in manifest query parameters: `:` ? & = `%` / (forward slash)

ADS query parameter length limitations

The following length limitations apply to query parameters that are use in requests to the ADS:

- **ADS parameter name**: 10000 characters
- **ADS parameter value**: 25000 characters
- **ADS URL**: 25000 characters

Advanced usage

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

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]
```

For more information about using dynamic domain, session, and player variables, select the applicable topic.

Topics

- Using domain variables to configure multiple content and ad sources
- Using session variables
- Using player variables

Using domain variables to configure multiple content and ad sources

With dynamic domain variables, you can use multiple domains, such as the **my-ads-server.com** part of the URL http://my-ads-server.com, with the player parameters in your configuration. This makes it possible for you to use more than one content source or ad decision server (ADS) in a single configuration.

You can use domain variables with any parameter that contains a URI:

- `AdDecisionServerUrl`
- `AdSegmentUrlPrefix`
- `ContentSegmentUrlPrefix`
- `LivePreroll.AdDecisionServerUrl`
- `VideoContentSourceUrl`

Domain variables are used alongside *configuration aliases* to perform dynamic variable replacement. Configuration aliases map a set of aliases and values to the player parameters that are used for dynamic domain configuration.

Topics

- Creating configuration aliases to use as dynamic variables
- Using configuration aliases to dynamically configure domains for a session

Creating configuration aliases to use as dynamic variables

Before you start to use domain variables, you create configuration aliases for your configuration. You use the configuration aliases as domain replacement variables at session initialization time. For example, you can use configuration aliases to dynamically configure an origin URL during session initialization.

Creating configuration aliases

To create configuration aliases to use for domain replacement using the MediaTailor console, perform the following procedure.

To create configuration aliases using the console

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. On the **Configuration aliases** section on the **Configurations** page, choose **Add player parameter**.
3. Type the player parameter name that you would like to use as a dynamic variable for domain replacement. You must prefix the name with `player_params.`.
4. Choose **OK**.

   AWS Elemental MediaTailor displays the new parameter in the table in the **Configuration aliases** section.
5. Now, you'll add an alias and value. Select the player parameter that you just named. This expands the section below the parameter name.

   Select **Add new alias**.
6. Enter an **Alias** key and **Value**. MediaTailor uses **Value** as the replacement value for the domain variable.

Using configuration aliases to dynamically configure domains for a session

After you set up the configuration aliases, you can use them as replacement variables for domains in your session initialization request. This enables you to dynamically configure the domains for your session.

Restrictions

Note the following restrictions when using configuration aliases:

- All dynamic variables used in the domain must be defined as a `ConfigurationAliases` dynamic variable.
- The player parameter variables must be prefixed with `player_params.`. For example, `player_params.origin_domain`.
- The list of aliased values must be exhaustive for every player parameter.
- If a request is made for a dynamic value that's used in the domain, and that request either doesn't specify the dynamic variable or one of the preconfigured aliases for that variable, then the request will fail with an HTTP `400` status code.

Example Example of usage

Here's an example of a configuration that includes configuration aliases and dynamic domain variables. Pay special attention to the player parameter variables, such as `[player_params.origin_domain]`, in the `AdDecisionServerUrl` and `VideoContentSourceUrl` parameter domains.

```
PUT /playbackConfiguration
   {
       "Name": "aliasedConfig",
       ...
       "AdDecisionServerUrl": "https://abc.execute-api.us-west-2.amazonaws.com/ads?
sid=[session.id]&ad_type=[player_params.ad_type]",
       "VideoContentSourceUrl": "https://[player_params.origin_domain].mediapackage.
[player_params.region].amazonaws.com/out/v1/[player_params.endpoint_id]",
       ...
       "ConfigurationAliases": {
           "player_params.origin_domain": {
               "pdx": "abc",
               "iad": "xyz"
           },
           "player_params.region": {
               "pdx": "us-west-2",
               "iad": "us-east-1"
           },
           "player_params.endpoint_id": {
               "pdx": "abcd",
               "iad": "wxyz"
           },
           "player_params.ad_type": {
               "customized": "abc12345",
```

```
                "default": "defaultAdType"
            },
        },
        ...
    }
```

Using the preceding configuration, create a session initialization request, specifying the player variables and aliases:

```
POST master.m3u8
   {
       "playerParams": {
           "origin_domain": "pdx",
           "region": "pdx",
           "endpoint_id": "pdx",
           "ad_type": "customized"
       }
   }
```

MediaTailor replaces the alias strings with the mapped values in the configuration aliases configuration.

The request to the ADS looks like this:

```
https://abc.execute-api.us-west-2.amazonaws.com/ads?sid=[session.id]&ad_type=abc12345
```

The request to the `VideoContentSource` looks like this:

```
https://777788889999.mediapackage.us-west-2.amazonaws.com/out/v1/abcd
```

Using session variables

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

The following table describes the session data variables you can use in your template ADS request URL configuration. The section numbers listed in the table correspond to the 2019a version of the

Society of Cable Telecommunications Engineers (SCTE)-35 specification, Digital Program Insertion Cueing Message, For details about ad prefetch, see Prefetching ads.

Name	Available for ad prefetch	SCTE-35 specification section	Description
[avail.index]	Yes		A number that represents the position of an ad avail in an index. At the start of a playback session, MediaTailor creates an index of all the ad avails in a manifest and stores the index for the remainder of the session. When MediaTailor makes a request to the ADS to fill the avail, it includes the ad avail index number. This parameter enables the ADS to improve ad selection by using features like competitive exclusion and frequency capping.
[avail.random]	Yes		A random number between 0 and 10,000,000,000, as a long number, that MediaTailor generates for each request to the ADS. Some ad servers use this parameter to enable features such as separating ads from competing companies.
[scte.archive_allowed_flag]	Yes	10.3.3.1	An optional Boolean value. When this value is 0, recording restrictions are asserted on the segment. When this value is 1, recording restrictions are not asserted on the segment.
[scte.avail_num]	Yes	9.7.2.1	The value parsed by MediaTailor from the SCTE-35 field `avail_num` , as a long number. MediaTailor can use this value to designate linear ad avail numbers. Value must be an integer.
[scte.avails_expected]	Yes	9,7.2.1	An optional long value that gives the expected count of avails within the current event.

Name	Available for ad prefetch	SCTE-35 specification section	Description
[scte.delivery_not_restricted_flag]	Yes	10.3.3.1	An optional Boolean value. When this value is 0, the next five bits are reserved. When this value is 1, the next five bits take on the meanings as described in the SCTE-35 specification.
[scte.device_restrictions]	Yes	10.3.3.1	An optional integer value that signals three pre-defined, independent, and non-hierarchical groups of devices. For more information about this variable, see the **segments_expected** description in the SCTE-35 specification.
[scte.event_id]	Yes	9.1 and 9.7.2.1	The value parsed by MediaTailor from the SCTE-35 field `splice_event_id` , as a long number. MediaTailor uses this value to designate linear ad avail numbers or to populate ad server query strings, like ad pod positions. Value must be an integer.
[scte.no_regional_blackout_flag]	Yes	10.3.3.1	An optional Boolean value. When this value is 0, regional blackout restrictions apply to the segment. When this value is 1, regional blackout restrictions do not apply to the segment.
[scte.segment_num]	Yes	10.3.3.1	An optional integer value that numbers segments within a collection of segments. For more information about this variable, see the **segment_num** description in the SCTE-35 specification.
[scte.segmentation_event_id]	Yes	10.3.3.1	MediaTailor exposes this variable as scte.event_id.

Name	Available for ad prefetch	SCTE-35 specification section	Description
`[scte.segmentation_type_id]`	Yes	10.3.3.1	An optional 8-bit integer value that specifies the segmentation type. For more information about this variable, see the **segmentation_type_id** description in the SCTE-35 specification.

Name	Available for ad prefetch	SCTE-35 specification section	Description
[scte.segmentation_upid]	segmentation_upid_type : Yes private_data : Yes	**segmentation_upid**: 10.3.3.1 Managed Private UPID: 10.3.3.3	Corresponds to the SCTE-35 segmentation_upid element. The segmentation_upid element contains segmentation_upid_type and segmentation_upid_length . MediaTailor supports the following segmentation_upid types: • **ADS Information (0x0E)** - Advertising information. For more information, see the **segmentation_upid** description in the SCTE-35 specification. • **Managed Private UPID (0x0C)** - The Managed Private UPID (MPU) structure as defined in the SCTE-35 specification. MediaTailor supports binary or DASH XML SCTE representations. You can use this structure in a podbuster workflow. To do so, specify a 32-bit (4 byte) format_identifier , and include the following parameters in the private_data attribute: `ABCD{"assetId":" my_program ","cueData": {"cueType":" theAdType ","key":" pb","value" :" 123456"}}` MediaTailor parses the values from the preceding JSON, and passes them into the scte.segmentation_upid.assetId , scte.segmentation_upid.cueData.key , and scte.segmentation_upid.cueData.value dynamic variables.

Name	Available for ad prefetch	SCTE-35 specification section	Description
			• **User Defined (0x01)** - A user defined structure. For more information, see the **segmentation_upid** description in the SCTE-35 specification.
[scte.segmentation_upid.assetId]	Yes		Used in conjunction with the Managed Private UPID (0xC) `segmentation_ upid_type` for podbuster workflows. MediaTailor derives this value from the `assetId` parameter in the MPU's `private_data` JSON structure. For more information, see Managed Private UPID JSON structure for a podbuster workflow.
[scte.segmentation_upid.cueData.key]	Yes		Used in conjunction with the Managed Private UPID (0xC) `segmentation_ upid_type` for podbuster workflows. MediaTailor derives this value from the `cueData.key` parameter in the MPU's `private_data` JSON structure. For more information, see Managed Private UPID JSON structure for a podbuster workflow.
[scte.segmentation_upid.cueData.value]	Yes		Used in conjunction with the Managed Private UPID (0xC) `segmentation_ upid_type` for podbuster workflows. MediaTailor derives this value from the `cueData.key` parameter in the MPU's `private_data` JSON structure. For more information, see Managed Private UPID JSON structure for a podbuster workflow. Value can be a string.

Name	Available for ad prefetch	SCTE-35 specification section	Description
[scte.segments_expected]	Yes	10.3.3.1	An optional integer value that gives the expected count of individual segments within a collection of segments. For more information about this variable, see the **segments_expected** description in the SCTE-35 specification.
[scte.sub_segment_num]	Yes	10.3.3.1	An optional integer value that identifies a particular sub-segment within a collection of sub-segments. For more information about this variable, see the **sub_segment_num** description in the SCTE-35 specification.
[scte.sub_segments_expected]	Yes	10.3.3.1	An optional integer value that gives the expected count of individual sub-segments within a collection of sub-segments. For more information about this variable, see the **sub_segments_expected** description in the SCTE-35 specification.
[scte.unique_program_id]	Yes	9.7.2.1	The integer value parsed by MediaTailor from the SCTE-35 `splice_insert` field `unique_program_id`. The ADS uses the unique program ID (UPID) to provide program-level ad targeting for live linear streams. If the SCTE-35 command is not splice insert, MediaTailor sets this to an empty value. Value must be an integer.

Name	Available for ad prefetch	SCTE-35 specification section	Description
[session.avail_duration_ms]	Yes		The duration in milliseconds of the ad availability slot. The default value is 300,000 ms. AWS Elemental MediaTailor obtains the duration value from the input manifest as follows: • For HLS: MediaTailor obtains the duration from the `#EXT-X-CUE-OUT: DURATION` or from values in the `#EXT-X-DATERANGE` tag. If the input manifest has a null, invalid, or 0 duration for the ad avail in those tags, MediaTailor uses the default. • For DASH: MediaTailor obtains the duration value from the event duration, if one is specified. Otherwise, it uses the default value. • For VOD: when a VOD stream triggers a pre-roll ad call, if the manifest does not include SCTE messaging with a duration value, MediaTailor does not enter a duration for the [session.avail_duration_ms], including the default duration value.
[session.avail_duration_secs]	Yes		The duration in seconds of the ad availability slot, or ad avail, rounded to the nearest second. MediaTailor determines this value the same way it determines `[session.avail_duration_ms]` .
[session.client_ip]	No		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` .

Name	Available for ad prefetch	SCTE-35 specification section	Description
[session.id]	No		A unique numeric identifier for the current playback session. All requests that a player makes for a session have the same id, so it can be used for ADS fields that are intended to correlate requests for a single viewing.
[session.referer]	No		Usually, the URL of the page that is hosting the video player. MediaTailor sets this variable to the value of the `Referer` header that the player used in its request to MediaTailor. If the player doesn't provide this header, MediaTailor leaves the **[session.referer]** empty. If you use a content delivery network (CDN) or proxy in front of the manifest endpoint and you want this variable to appear, proxy the correct header from the player here.
[session.user_agent]	No		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]	No		Alternative to **[session.id]** . This is a unique identifier for the current playback session, such as the following: `e039fd39-09f0-46b2-aca9-9871cc116cde`

Example

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
```

If the ADS requires several query parameters to be passed, the template ADS URL in AWS Elemental MediaTailor could look like the following:

```
https://my.ads.server.com/sample?
e=[scte.avails_expected]&f=[scte.segment_num]&g=[scte.segments_expected]&h=[scte.sub_segment_nu
```

The following DASH marker example XML fragment shows how to use `scte35:SpliceInsert`:

```
<Period start="PT444806.040S" id="123456" duration="PT15.000S">
  <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
    <Event duration="1350000">
      <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="180832" tier="4095">
        <scte35:SpliceInsert spliceEventId="1234567890"
 spliceEventCancelIndicator="false" outOfNetworkIndicator="true"
 spliceImmediateFlag="false" uniqueProgramId="1" availNum="1" availsExpected="1">
          <scte35:Program><scte35:SpliceTime ptsTime="5672624400"/></scte35:Program>
          <scte35:BreakDuration autoReturn="true" duration="1350000"/>
        </scte35:SpliceInsert>
      </scte35:SpliceInfoSection>
```

The following DASH marker example XML fragment shows how to use `scte35:TimeSignal`:

```
<Period start="PT346530.250S" id="123456" duration="PT61.561S">
  <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
    <Event duration="5310000">
      <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="183003" tier="4095">
        <scte35:TimeSignal>
```

```
          <scte35:SpliceTime ptsTime="3442857000"/>
        </scte35:TimeSignal>
        <scte35:SegmentationDescriptor segmentationEventId="1234567"
 segmentationEventCancelIndicator="false" segmentationDuration="8100000"
 segmentationTypeId="52" segmentNum="0" segmentsExpected="0">
          <scte35:DeliveryRestrictions webDeliveryAllowedFlag="false"
 noRegionalBlackoutFlag="false" archiveAllowedFlag="false" deviceRestrictions="3"/>
          <scte35:SegmentationUpid segmentationUpidType="12"
 segmentationUpidLength="2">0100</scte35:SegmentationUpid>
        </scte35:SegmentationDescriptor>
      </scte35:SpliceInfoSection>
    </Event>
```

The following DASH marker example XML fragment shows how to use `scte35:Binary`:

```
<Period start="PT444806.040S" id="123456" duration="PT15.000S">
  <EventStream schemeIdUri="urn:scte:scte35:2014:xml+bin" timescale="1">
    <Event presentationTime="1541436240" duration="24" id="29">
      <scte35:Signal xmlns="http://www.scte.org/schemas/35/2016">
        <scte35:Binary>/DAhAAAAAAAAAP/wEAUAAAHAf+9/fgAg9YDAAAAAAAA25aoh</Binary>
      </scte35:Signal>
    </Event>
    <Event presentationTime="1541436360" duration="24" id="30">
      <scte35:Signal xmlns="http://www.scte.org/schemas/35/2016">

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

The following HLS tag example shows how to use `EXT-X-DATERANGE`:

```
#EXT-X-DATERANGE:ID="splice-6FFFFFF0",START-DATE="2014-03-05T11:
15:00Z",PLANNED-DURATION=59.993,SCTE35-OUT=0xFC002F0000000000FF0
00014056FFFFFF000E011622DCAFF000052636200000000000A0008029896F50
000008700000000
```

The following HLS tag example shows how to use `EXT-X-CUE-OUT`:

```
#EXT-OATCLS-SCTE35:/DA0AAAAAAAAAAAABQb+ADAQ6QAeAhxDVUVJQAAAO3/PAAEUrEoICAAAAAAg
+2UBNAAANvrtoQ==
#EXT-X-ASSET:CAID=0x0000000020FB6501
```

```
#EXT-X-CUE-OUT:201.467
```

The following HLS tag example shows how to use `EXT-X-SPLICEPOINT-SCTE35`:

```
#EXT-X-SPLICEPOINT-SCTE35:/DA9AAAAAAAAAP/wBQb+uYbZqwAnAiVDVUVJAAAKqX//
AAEjW4AMEU1EU05CMDAxMTMyMjE5M19ONAAAmXz5JA==
```

The following example shows how to use `scte35:Binary` decode:

```
{
  "table_id": 252,
  "section_syntax_indicator": false,
  "private_indicator": false,
  "section_length": 33,
  "protocol_version": 0,
  "encrypted_packet": false,
  "encryption_algorithm": 0,
  "pts_adjustment": 0,
  "cw_index": 0,
  "tier": "0xFFF",
  "splice_command_length": 16,
  "splice_command_type": 5,
  "splice_command": {
    "splice_event_id": 448,
    "splice_event_cancel_indicator": false,
    "out_of_network_indicator": true,
    "program_splice_flag": true,
    "duration_flag": true,
    "splice_immediate_flag": false,
    "utc_splice_time": {
      "time_specified_flag": false,
      "pts_time": null
    },
    "component_count": 0,
    "components": null,
    "break_duration": {
      "auto_return": false,
      "duration": {
        "pts_time": 2160000,
        "wall_clock_seconds": 24.0,
        "wall_clock_time": "00:00:24:00000"
      }
    },
```

```
    "unique_program_id": 49152,
    "avail_num": 0,
    "avails_expected": 0
    "segment_num": 0,
    "segments_expected": 0,
    "sub_segment_num": 0,
    "sub_segments_expected": 0
  },
  "splice_descriptor_loop_length": 0,
  "splice_descriptors": null,
  "Scte35Exception": {
    "parse_status": "SCTE-35 cue parsing completed with 0 errors.",
    "error_messages": [],
    "table_id": 252,
    "splice_command_type": 5
  }
}
```

Using player variables

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

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

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

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

To add query parameters as key-value pairs

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

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

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

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

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

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

   - Example requests for server-side ad-tracking reporting - using URL-encoded pairs

     HLS:

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

     DASH:

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

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

     HLS:

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

     DASH:

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

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

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

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

Passing AWS Elemental MediaTailor session initialization parameters to the manifest

AWS Elemental MediaTailor can preserve query parameters from session initialization and append them to the personalized manifest URL returned to the client player. Subsequent client requests also contain the appended query parameters.

Manifest query parameters are useful if you use a Content Delivery Network (CDN) between MediaTailor and the client player, where the CDN uses the query parameters for the following:

- Dynamic routing to different MediaTailor endpoints
- Token authorization

For client-side reporting, MediaTailor appends query parameters for client-side reporting endpoints, but it doesn't append the query parameters for the CloudFront (or other CDN) segments.

To use parameter preservation, submit an AWS Support ticket to request for manifest query parameter pass through to be enabled.

The behavior varies between HLS and DASH, as well as with explicit and implicit session initialization. The following topics describe how to configure session initialization requests so MediaTailor will pass through parameters to the manifest.

Topics

- HLS implicit session initialization with AWS Elemental MediaTailor
- DASH implicit session initialization with AWS Elemental MediaTailor
- HLS and DASH explicit session initialization with AWS Elemental MediaTailor

HLS implicit session initialization with AWS Elemental MediaTailor

When the request includes query parameters with the key `manifest.*`, as shown in the following example, MediaTailor includes the query parameters in the links to MediaTailor resources. Links don't include the `manifest.` prefix.

```
GET /v1/master/111122223333/originId/index.m3u8?manifest.test=123&other=456
```

Example parent manifest

In the following example, MediaTailor includes the query parameters MediaTailor for URL for the parent manifest.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-MEDIA:LANGUAGE="eng",AUTOSELECT=YES,FORCED=NO,TYPE=SUBTITLES,URI="../../../
manifest/111122223333/originId/session/1.m3u8?manifest.test=123",GROUP-
ID="subtitles",DEFAULT=YES,NAME="caption_1"
#EXT-X-STREAM-INF:CODECS="avc1.640029,mp4a.40.2",AVERAGE-
BANDWIDTH=2525657,RESOLUTION=960x540,SUBTITLES="subtitles",FRAME-
RATE=29.97,BANDWIDTH=2665212
../../../manifest/111122223333/originId/session/0.m3u8?manifest.test=123
```

Example child manifest

In the following example, MediaTailor includes the query parameters in the URLs for the content segments.

```
#EXTM3U
```

```
#EXT-X-VERSION:6
#EXT-X-TARGETDURATION:7
#EXT-X-MEDIA-SEQUENCE:28716269
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXTINF:6.006,
https://origin.com/contentSegment_1.ts?originQueryParam=foo
#EXT-X-DISCONTINUITY
#EXTINF:6.006,
../../../../segment/111122223333/originId/session/0/2?manifest.test=123
```

DASH implicit session initialization with AWS Elemental MediaTailor

The client makes a manifest request without a session, as shown in the following example.

```
GET /v1/dash/111122223333/originId/index.mpd?manifest.test=123&other=456
```

MediaTailor creates a session for the client and redirects it with the query parameters:

```
/v1/dash/111122223333/originId/index.mpd?sessionId=session&manifest.test=123
```

When the client makes the request, MediaTailor responds with a DASH manifest similar to the following example. The first period is a content period, so MediaTailor doesn't insert the manifest query parameter there. In the second period, which is an ad period, MediaTailor inserts the manifest query parameter into the `SegmentTemplate` element's `initialization` attribute and `media` attribute. The `Location` element also has the manifest query parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD availabilityStartTime="2018-07-27T09:48:23.634000+00:00"
 id="201" minBufferTime="PT30S" minimumUpdatePeriod="PT15S"
 profiles="urn:mpeg:dash:profile:isoff-live:2011" publishTime="2023-02-14T23:37:43"
 suggestedPresentationDelay="PT25.000S" timeShiftBufferDepth="PT56.997S" type="dynamic"
 xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:scte35="urn:scte:scte35:2013:xml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/ittf/
PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd">
    <BaseURL>https://origin.com/contentSegments/</BaseURL>
    <Location>https://mediatailor.com/v1/dash/111122223333/originId/index.mpd?
manifest.test=123&aws.sessionId=session</Location>
    <Period duration="PT29.963S" id="28737823" start="PT143732873.178S">
```

```
        <AdaptationSet bitstreamSwitching="true" mimeType="video/mp4"
 segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
            <Representation bandwidth="2200000" codecs="avc1.640029"
 frameRate="30000/1001" height="540" id="1" width="960">
                <SegmentTemplate initialization="index_video_7_0_init.mp4?
m=1611174111" media="index_video_7_0_$Number$.mp4?m=1611174111"
 presentationTimeOffset="4311986195351" startNumber="28737828" timescale="30000">
                    <SegmentTimeline>
                        <S d="180180" t="4311986911066"/>
                        <S d="3003" t="4311987091246"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
    <Period id="28737829_1" start="PT39925H48M23.141S">
        <BaseURL>https://mediatailor.com/v1/
dashsegment/111122223333/originId/session/28737829/28737829_1/</BaseURL>
        <AdaptationSet bitstreamSwitching="false" frameRate="30000/1001"
 mimeType="video/mp4" segmentAlignment="true" startWithSAP="1"
 subsegmentAlignment="true" subsegmentStartsWithSAP="1">
            <SegmentTemplate startNumber="1" timescale="90000"/>
            <Representation bandwidth="2200000" codecs="avc1.64001f" height="540"
 id="1" width="960">
                <SegmentTemplate initialization="asset_540_2_0init.mp4?
manifest.test=123" media="asset_540_2_0_$Number%09d$.mp4?manifest.test=123"
 startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180180" r="6" t="0"/>
                        <S d="87087" t="1261260"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
</MPD>
```

HLS and DASH explicit session initialization with AWS Elemental MediaTailor

When the client makes an explicit session initialization request, MediaTailor includes the `manifestParams` as query parameters in the parent manifest and tracking URLs in the response.

Example session initialization request

```
POST /v1/session/111122223333/originId/index.m3u8
{
    "adsParams": {
        "param1": "value1",
        "param2": "value2",
        "param3": "value3"
    },
    "manifestParams": {
        "test": "123"
    }
}
```

Example manifest and tracking response

```
{
    "manifestUrl": "/v1/master/111122223333/originId/index.m3u8?
aws.sessionId=session&test=123",
    "trackingUrl": "/v1/tracking/111122223333/originId/session?test=123"
}
```

Manifest responses for the session have the specific `manifestParams` in MediaTailor URLs similar to the previously described implicit session-initialization workflows. The key difference is that the manifest parameters for explicit session initialization don't start with `manifest.`.

Manifest query parameters are immutable and only set on session initialization. If a client makes multiple parent manifest requests for a single session, MediaTailor doesn't update the manifest query parameters after the first request.

Reporting ad tracking data

MediaTailor provides two options for tracking and reporting on how much of an ad a viewer has watched. In the server-side ad reporting approach, MediaTailor tracks the ad and sends beacons

(tracking signals) directly to the ad server. Alternatively, in the client-side tracking approach, the client player (the user's device) tracks the ad and sends the beacons to the ad server. The type of ad reporting used in a playback session depends on the specific request the player makes to initiate the session in MediaTailor.

Topics

- Server-side ad tracking
- Client-side ad tracking

Server-side ad tracking

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

To perform server-side ad reporting

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

  - Example: HLS format

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

  - Example: DASH format

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

  The key-value pairs are the dynamic targeting parameters for ad tracking. For information about adding parameters to the request, see the section called “Using dynamic ad variables”.

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

> **Note**
>
> When MediaTailor encounters a double-slash (//) in a tracking URL, it collapses the slashes to one (/).

When the player requests playback from an ad segment URL (`/v1/segment` path), AWS Elemental MediaTailor sends the appropriate beacon to the ad server through the ad tracking URLs. At the same time, the service issues a redirect to the actual `*.ts` ad segment. The ad segment is either in the Amazon CloudFront distribution where MediaTailor stores transcoded ads, or in the content distribution network (CDN) where you have cached the ad.

Client-side ad tracking

Using the AWS Elemental MediaTailor client-side tracking API, you can incorporate player controls during ad breaks in streaming workflows. In client-side tracking, the player or client emits tracking events, such as impression and quartile ad beaconing, to the Ad Decision Server (ADS) and other ad-verification entities. For more information about impression and quartile ad beaconing, see Client-side beaconing. For more information about ADS and other ad-verification entities, see Client-side ad-tracking integrations.

Client-side tracking enables functionality like the following:

- Ad-break countdown timers - For more information, see Ad countdown timer.
- Ad click-through - For more information, see Ad click-through.
- Display of companion ads - For more information, see Companion ads.
- Skippable ads - For more information, see Skippable ads.
- Display of VAST icons for privacy compliance - For more information, see Icons for Google Why This Ad (WTA).
- Control of player scrubbing during ads - For more information, see Scrubbing.

Using the MediaTailor client-side tracking API, you can send metadata to the playback device that enables functionality in addition to client-side tracking:

Topics

- Enabling client-side tracking
- Ad server parameters
- Origin interaction query parameters
- Session-configured features
- Best practices for client-side tracking
- Client-side ad-tracking schema and properties
- Ad-tracking activity timing
- Player controls and functionality for client-side ad tracking
- Client-side beaconing
- Hybrid mode with server-side ad beacons
- Client-side ad-tracking integrations
- Paging through ad beacons with GetTracking

Enabling client-side tracking

You enable client-side tracking for each session. The player makes an HTTP POST to the MediaTailor configuration's session-initialization prefix endpoint. Optionally, the player can send additional metadata for MediaTailor to use when making ad calls, calling the origin for a manifest, and invoking or disabling MediaTailor features at the session level.

The following example shows the structure of the JSON metadata:

```
{
  "adsParams": {                  # 'adsParams' is case sensitive
    "param1": "value1",           # key is not case sensitive
    "param2": "value2",           # Values can contain spaces. For example, 'value 2'
 is an allowed value.
    },
  "origin_access_token":"abc123", # this is an example of a query parameter designated
 for the origin
  "overlayAvails":"on"            # 'overlayAvails' is case sensitive. This is an
 example of a feature that is enabled at the session level.
}
```

Use the MediaTailor console or API to configure the ADS request template URL to reference these parameters. In the following example, `player_params.param1` are the player parameters for `param1`, and `player_params.param2` are the player parameters for `param2`.

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

Ad server parameters

At the topmost level of the JSON structure is an `adsParams` JSON object. Inside this object are key/value pairs that MediaTailor can read and send to the ad server in all session requests. MediaTailor supports the following ad servers:

- Google Ad Manager
- SpringServe
- FreeWheel
- Publica

Origin interaction query parameters

Any reserved key/value pairs within the topmost level of the JSON structure, such as `adParams`, `availSuppression`, and `overlayAvails`, aren't added to the origin request URL in the form of query parameters. Every session manifest request that MediaTailor makes to the origin contains these query parameters. The origin ignores extraneous query parameters. For example, MediaTailor can use the key/value pairs to send access tokens to the origin.

Session-configured features

Use the session-initialization JSON structure to enable, disable, or override MediaTailor features such as `overlayAvails`, `availSuppression`, and `adSignaling`. Any feature configurations passed during session initialization override the setting at the MediaTailor configuration level.

> **Note**
>
> The metadata submitted to MediaTailor at session initialization is immutable, and additional metadata cannot be added for the duration of the session. Use SCTE-35 markers to carry data that changes during the session. For more information, see Using session variables.

Example : Performing client-side ad tracking for HLS

```
POST mediatailorURL/v1/session/hashed-account-id/origin-id/asset-id.m3u8

        {
            "adsParams": {
               "deviceType": "ipad"   # This value does not change during the session.
               "uid": "abdgfdyei-2283004-ueu"
           }
        }
```

Example : Performing client-side ad tracking for DASH

```
POST mediatailorURL/v1/session/hashed-account-id/origin-id/asset-id.mpd

        {
            "adsParams": {
               "deviceType": "androidmobile",
               "uid": "xjhhddli-9189901-uic"
           }
        }
```

A successful response is an HTTP 200 with a response body. The body contains a JSON object with a `manifestUrl` and a `trackingUrl` key. The values are relative URLs that the player can use for both playback and ad-event tracking purposes.

```
{
  "manifestUrl": "/v1/dashmaster/hashed-account-id/origin-id/asset-id.m3u8?
aws.sessionId=session-id",
  "trackingUrl": "/v1/tracking/hashed-account-id/origin-id/session-id"
}
```

For more information on the client-side tracking schema, see Client-side ad-tracking schema and properties.

Best practices for client-side tracking

This section outlines the best practices for client-side tracking in MediaTailor for both live and VOD workflows.

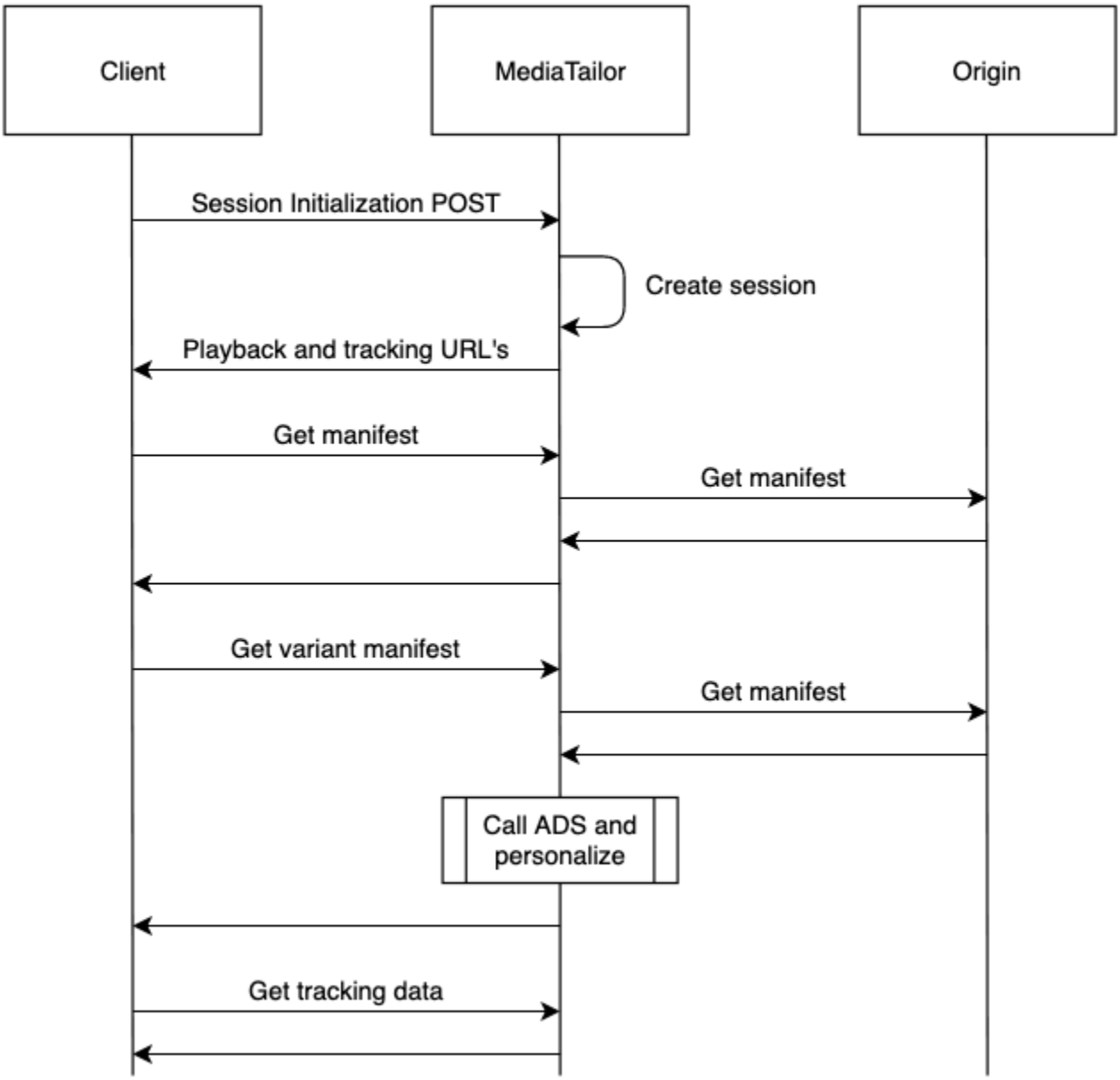
Live workflows

Poll the tracking endpoint at an interval matching every target duration for HLS, or minimum update period for DASH, in order to always have the most current ad-tracking metadata. Matching this interval is especially important in workflows where the creatives might have an interactive or overlay component.

> **Note**
>
> Some players support event listeners, which could be used as an alternative to polling. For example, the MediaTailor ad ID decoration feature would need to be enabled for each session. For more information, see Ad ID decoration. Using this feature puts a date range (HLS) or event element (DASH) identifier over each ad in the avail. Players can use these manifest tags as a prompt to call the MediaTailor tracking endpoint for the session.

VOD workflows

Following a successful session initialization, and after MediaTailor receives the first manifest containing media, you only need to call the tracking endpoint once.



Client-side ad-tracking schema and properties

With the MediaTailor client-side ad-tracking feature, you can integrate detailed client-side ad tracking data into your player environment. The following sections cover the overall ad-tracking schema, as well as the specific properties and values that make up the schema.

Contents

- Schema
- Properties

Schema

The following table describes the MediaTailor client-side ad-tracking schema. Where applicable, the table maps the schema to VAST data.

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
Response JSON					
-	Object	avails, nonLinearAvails			
`/avails`	Array				
MediaTailor creates one object for each avail (ad break) inside the manifest window.		ads, adType, availID, duration, durationInSeconds, startTime, startTimeInSeconds, dateTime			
`/ads`	Array				
MediaTailor creates one object for each ad inside the avail period.	Object	adID, adType, adParameters, adVerifications, companionAds, duration, durationInSeconds, extension			

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
		s, icons, mediaFile, startTime, startTimeInSeconds, dateTime, adBreakTrackingEvents			
/adId	String				• HLS - the sequence number associated with the beginning of the ad • DASH - the period ID of the ad
/adParameters	String		VAST/Ad/InLine/Creatives/Creative/Linear/AdParameters		String of ad parameters from the VAST VPAID that MediaTailor passes to the player

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/adVerifications	Array	VAST/Ad/InLine/AdVerifications			Contains the resources and metadata required to execute third-party measurement code in order to verify creative playback
MediaTailor creates an object for each ad-verification element.	Object	executableResource, javaScriptResource, vendor, verificationParameters			
/executableResource	Array		VAST/Ad/InLine/AdVerifications/Verification/ExecutableResource		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
MediaTailor creates an object for each `executableResource` element.	Object	apiFramework, adType, uri, language			
`/apiFramework`	String		`VAST/Ad/InLine/AdVerifications/Verification/ExecutableResource/@apiFramework`		
`/type`	String				
`/uri`	String		`VAST/Ad/InLine/AdVerifications/Verification/ExecutableResource/#CDATA`		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
`/language`	String		`VAST/Ad/InLine/AdVerifications/Verification/ExecutableResource/@language`	`VAST/Ad/InLine/AdVerifications/Verification/ExecutableResource/@language`	
`/javaScriptResource`	Array		`VAST/Ad/InLine/AdVerifications/Verification/JavaScriptResource`		
MediaTailor creates an object for each `javaScriptResource` element.	Object	apiFramework, browserOptional, uri			

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/apiFramework	String		VAST/Ad/InLine/AdVerifications/Verification/JavaScriptResource/@apiFramework		
/browserOptional	String		VAST/Ad/InLine/AdVerifications/Verification/JavaScriptResource/@browserOptional		
/uri	String		VAST/Ad/InLine/AdVerifications/Verification/JavaScriptResource/#CDATA		
/trackingEvents	Array				

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
MediaTailor creates an object for each ad-verification element tracking-event type.	Object	event, uri			
/event	String		VAST/Ad/InLine/AdVerifications/Verification/TrackingEvents/Tracking/@event		
/uri	String		VAST/Ad/InLine/AdVerifications/Verification/TrackingEvents/Tracking/#CDATA		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/vendor	String		VAST/Ad/InLine/AdVerifications/Verification/@vendor		
/verificationParameters	String		VAST/Ad/InLine/AdVerifications/Verification/VerificationParameters		
/companionAds	Array				Companion ads, which accompany the ad avail, provide content like a frame around the ad or a banner to display near the video.

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
MediaTailor creates an object for each companion-ad element.	Object	adParameters, altText, attributes, companionClickThrough, companionClickTracking, htmlResource, sequence, staticResource, trackingEvents	`VAST/Ad/InLine/Creatives/Creative/CompanionAds`		
`/adParameters`	String				
`/altText`	String				

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/attributes	Object	adSlotId, apiFramework, assetHeight, assetWidth, expandedHeight, expandedWidth, height, id, pxratio, renderingMode, width			
/adSlotId	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/@adSlotId		
/apiFramework	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/@apiFramework		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/assetHeight	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/@assetHeight		
/assetWidth	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/@assetWidth		
/expandedHeight	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/@expandedHeight		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/expandedWidth	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/@expandedWidth		
/height	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/@height		
/id	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/@id		
/pxratio	String				
/renderingMode	String				

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/width	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/@width		
/companionClickThrough	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/CompanionClickThrough		
/companionClickTracking	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/CompanionClickTracking		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/htmlResource	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/HTMLResource		
/iFrameResource	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/iFrameResource		
/sequence	String				
/staticResource	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/StaticResource		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/trackingEvents	Array		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/TrackingEvents		
MediaTailor creates an object for each companion-ad element-tracking event type.					
/tracking	Object	>event, uri			
/event	String		VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/TrackingEvents/Tracking/@event		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
`/uri`	String		`VAST/Ad/InLine/Creatives/Creative/CompanionAds/Companion/TrackingEvents/Tracking/#CDATA`		
`/duration`	String				Length, in ISO 8601 seconds format
`/durationInSeconds`	Number				Length, in seconds format
`/extensions`	Array				Ad servers can use custom VAST extensions
MediaTailor creates an object for each child extension of the `extensions` element.			`VAST/Ad/InLine/Extensions`		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/extension	Object	type, content	VAST/Ad/InLine/Extensions/Extension		
/type	String		VAST/Ad/InLine/Extensions/Extension/@type		
/content	String				
/icons	Array		VAST/Ad/InLine/Creatives/Creative/Linear/Icons		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
MediaTailor creates an object for each `icon` element within `icons`.	Object	attributes, dateTime, duration, durationInSeconds, htmlResource, iconClicks, iconViewTracking, iFrameResource, staticResource, startTime, startTimeInSeconds	`VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon`		
`/attributes`	Object	apiFramework, duration, height, offset, program, pxratio, width, xPosition, yPosition			

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/apiFramework	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/@apiFramework		
/duration	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/@duration		
/height	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/@height		
/offset	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/@offset		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/program	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/@program		
/pxratio	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/@pxratio		
/width	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/@width		
/xPosition	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/@xPosition		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/yPosition	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/@yPosition		
/dateTime	String				
/duration	String				
/durationInSeconds	Number				
/htmlResource	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/HTMLResource		
/iconClicks	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/IconClicks		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/iconClickThrough	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/IconClicks/IconClickThrough		
/iconClickTracking	Object	id	VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/IconClicks/IconClickTracking		
/id	String				
/iconClickFallbackImages	Array		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/IconClicks/IconClickFallbackImages		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
MediaTailor creates an object for each icon-click fallback-image node.					
/altText	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/IconClicks/IconClickFallbackImages/IconClickFallbackImage/AltText		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/height	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/IconClicks/IconClickFallbackImages/IconClickFallbackImage/@height		
/width	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/IconClicks/IconClickFallbackImages/IconClickFallbackImage/@width		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/staticResource	Object	creativeType, uri	VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/IconClicks/IconClickFallbackImages/IconClickFallbackImage/StaticResource		
/creativeType	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/IconClicks/IconClickFallbackImages/IconClickFallbackImage/StaticResource/@creativeType		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/uri	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/IconClicks/IconClickFallbackImages/IconClickFallbackImage/StaticResource/#CDATA		
/iconViewTracking	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/IconViewTracking		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/iFrameResource	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/iFrameResource		
/statidResource	Object	creativeType, uri	VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/StaticResource		
/creativeType	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/StaticResource/@type		

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/uri	String		VAST/Ad/InLine/Creatives/Creative/Linear/Icons/Icon/StaticResource/#CDATA		
/startTime	String				
/startTimeInSeconds	Number				
/mediaFiles	Object	adParameters, duration, durationInSeconds, mediaFilesList, mezzanine, startTime, startTimeInSeconds, trackingEvents			Video and other assets that the player needs for the ad avail
/adParameters	String				

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/duration	String				
/durationInSeconds	Number				
/mediaFilesList	Array				
MediaTailor creates an object for each companion-ad element-tracking event type		apiFramework, delivery, height, maintainAspectRatio, mediaFileUri, mediaType, scalable, width			
/apiFramework	String				
/delivery	String				
/height	String				
/maintainAspectRatio	String				
/mediaFileUri	String				
/mediaType	String				

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/scalable	String				
/width	String				
/mezzanine	String				
/startTime	String				
/startTimeInSeconds	String				
/trackingEvents	Array				
MediaTailor creates an object for each tracking event for the creative		beaconUrls, duration, durationInSeconds, dateTime, eventId, eventType, startTime, startTimeInSeconds			
/beaconUrls	Array				

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
A comma-separated list of all tracking URLs for this event					
/duration	String				
/durationInSeconds	Number				
/dateTime	String				
/eventId	String				
/eventType	String				
/startTime	String				
/startTimeInSeconds	Number				

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/startTime	String				Time position, in ISO 8601 seconds format, relative to the beginning of the playback session
/startTimeInSeconds	Number				Time position, in seconds format, relative to the beginning of the playback session
/dateTime	String				Program date time, in ISO 8601 seconds format, for the start of the ad avail

Key	Data type	Child keys	Mapping from VAST 2.0/3.0	Mapping from VAST 4.0	Description
/trackingEvents	Array				Contains all tracking-event data that was received in the VAST response, along with timing information
/adType	String				
/availId	String				
/dateTime	String				
/duration	String				
/durationInSeconds	Number				
/startTime	String				
/startTimeInSeconds	Number				

Properties

The following table lists the properties in the client-side tracking API, their definitions, value types, and examples.

Property	Definition	Value type	Example
`adID`	• HLS - the sequence number associated with the beginning of the ad • DASH - the period ID of the ad	String	`10`
`adBreakTrackingEvents`	An array that carries VMAP tracking events from the VAST response. For more information, see section 2.3.3 of the VMAP 1.0 specification.	String	`[]`
`adMarkerDuration`	The avail duration observed from the ad marker in the manifest.	String	`30`
`adParameters`	A string of ad parameters, from the VAST VPAID, that MediaTailor passes to the player.	String	
`adProgramDateTime`	• HLS - the date, in ISO/IEC 8601:2004 format, that represents the first media sequence of the ad. • DASH -	String	

Property	Definition	Value type	Example
`ads`	An array containing the ad objects that make up the avail. The ads are listed in the order they appear in the manifest.	Array	`[]`
`adSystem`	The name of the system that serves the ad. **Important** Make sure to provide a value. If you don't provide a value, issues can arise.	String	`myADS`
`adTitle`	The title of the ad.	String	`ad1`

Property	Definition	Value type	Example
`adVerifications`	Contains the resources and metadata required to execute third-party measurement code in order to verify creative playback. For more information about this property, see section 3.16 of the VAST 4.2 specification. MediaTailor supports `adVerifications` as VAST 3 extension nodes.	Array	`[]`
`altText`	The alternative text for an image of a companion ad. This text enables players with descriptive-audio support for the visually impaired to read back a description of the image.	String	`video sequence advertising sneakers`
`attributes`	Includes keys defined in the VAST specification for companion ads, such as `adSlotId`, `pxratio`, `renderingMode` , and so on.	Object	`{}`

Property	Definition	Value type	Example
`apiFramework`	Set to `VPAID` to tell the player that this ad is a VPAID ad.	String	`VPAID`
`availID`	• HLS - the sequence number associated with the start of the ad avail. • DASH - the period ID of the ad avail, which is usually the period ID of the content that is to be replaced with an ad.	String	• `34` • `PT34S_1`
`avails`	An array containing ad-break objects, or *avails*, that are presented in the active manifest window. The avails are listed in the order they appear in the manifest.	Array	`[]`
`beaconUrls`	The URL where MediaTailor sends the ad beacon.	String	
`bitrate`	The bitrate of the video asset. This property is not typically included for an executable asset.	String	`2048`

Property	Definition	Value type	Example
companionAds	One or more companion ad-content specifications, each of which specifies a resource file to use. Companion ads accompany the ad avail and provide content, like a frame around the ad or a banner, to display near the video.	Array	[]
companionClickThrough	A URL to the advertiser's page that the media player opens when the viewer clicks the companion ad.	String	https://aws.amazon.com/
companionClickTracking	The tracking URL for the companionClickThrough property.	String	https://myads.com/beaconing/event=clicktracking
creativeId	The Id attribute value of the Creative tag for the ad.	String	creative-1

Property	Definition	Value type	Example
creativeSequence	The sequence in which an ad should play, according to the Ad@id value in the VAST response.	String	1
dashAvailabilityStartTime	For live/dynamic DASH, the MPD@availabilityStartTime of the origin manifest.	String	2022-10-05T19:38:39.263Z
delivery	Indicates whether a progressive or streaming protocol is being used.	String	progressive
duration	Length, in ISO 8601 seconds format. The response includes durations for the entire ad avail and for each ad and beacon, though beacon durations are always zero.	Number	15.015
eventId	• HLS - the sequence number associated with the beacon. • DASH - the ptsTime of the start of the ad.	String	23

Property	Definition	Value type	Example
`eventType`	The type of beacon.	String	`impression`
`extensions`	Custom extensions of VAST that ad servers use. For more information about extensions, see section 3.18 of the VAST 4.2 specification.	Array	`[]`
`height`	The height, in pixels, of the video asset.	String	`360`
`hlsAnchorMediaSequenceNumber`	The media-sequence number of the first/oldest media sequence seen in the HLS origin manifest.	String	`77`
`htmlResource`	The CDATA-encoded HTML that's inserted directly within the streaming provider's HTML page.	String	`<![CDATA[ <!doctype html><html><head><meta name=\"viewport\" content=\"width=1, initial-scale=1.0, minimum-scale=1.0, ...]]>`

Property	Definition	Value type	Example
iFrameResource	The URL to an HTML resource file that the streaming provider loads into an iframe.	String	
maintainAspectRatio	Indicates whether to maintain the video's aspect ratio while scaling.	Boolean	true
mediaFilesList	Specifies the video and other assets that the player needs for the ad avail.	Array	[]
mediaFileUri	URI that points to either an executable asset or a video asset.	String	https://myad.com/ad/ad134/vpaid.js
mediaType	The MIME type of the creative or companion asset.	String	video/mp4
meta			
mezzanine	The URL of the mezzanine MP4 asset, specified if the VPAID ad includes one.	String	https://gcdn.2mdn.net/videoplayback/id/itag/ck2/file/file.mp4

Property	Definition	Value type	Example
nextToken	The value of the token that points to the next page of results, when such a value exists.	String	UFQzOS44NzNTXzIwMjMtMDctMzFUMTY6NTA6MDYuMzUwNjI2ODQ1Wl8x
nonLinearAds		Array	[]
nonLinearAdsList		Array	[]
nonLinearAvails		Array	
scalable	Indicates whether to scale the video to other dimensions.	Boolean	true
sequence	The sequence value specified for the creative in the VAST response.	String	1
skipOffset	The time value that identifies when the player makes skip controls available to the user.	String	00:00:05

Property	Definition	Value type	Example
`startTime`	The time position, in ISO 8601 seconds format, relative to the beginning of the playback session. The response includes start times for the entire ad avail and for each ad and beacon.	String	`PT9.943S`
`startTime InSeconds`	The time position, in seconds format, relative to the beginning of the playback session. The response includes start times for the entire ad avail and for each ad and beacon.	Number	`9.943`
`staticResource`	The URL to a static creative file that's used for the ad component.	String	`https://very-interactive-ads.com/campaign1/file.json?c=1019113602`
`vastAdId`	The `Id` attribute value of the `Ad` tag.	String	`ad1`
`width`	The width, in pixels, of the video asset.	String	`640`

Ad-tracking activity timing

With client-side reporting, the player must emit tracking events (beacons) with a level of precision. Using the MediaTailor client-side tracking schema, you can ensure that, for every avail, ad, companion, overlay, and tracking events, timing and duration information is present, and in different forms.

Use the following MediaTailor key/value pairs for the player to accurately reconcile ad-event activities, such as tracking events, with playback position:

- startTime
- startTimeInSeconds
- adProgramDateTime
- adID/eventId

HLS and DASH implement the value of `startTime` and `startTimeInSeconds` differently:

- HLS - The `startTime` values are relative to the beginning of the playback session. The beginning of the playback session is defined as time zero. The ad's `startTime` is the sum of the cumulative values of all the `EXT-INF` segment durations leading up to the avail. The media-sequence number of the segment that the ad or tracking event falls on also corresponds to the `adId` or `eventId` in the client-side tracking response.
- DASH:
  - Live/dynamic manifests - The `startTime` values are relative to the `MPD@availabilityStartTime` of the DASH manifest. The `MPD@avaibilityStartTime` is a timing anchor for all MediaTailor sessions that consume the stream.
  - VOD/static manifests - The `startTime` values are relative to the beginning of the playback session. The beginning of the playback session is defined as time zero. Each ad inside the avail is contained inside its own `Period` element. The `Period` element has a `@start` attribute with a value that's the same as the `startTime` values in the client-side tracking payload. The `PeriodId` also corresponds to the `adId` or `eventId` in the client-side tracking response.

Example : HLS

In the following example, the MediaTailor session started, and the following manifest is the first one served to the client:

```
#EXTM3U
#EXT-X-VERSION:6
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:4603263
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-PROGRAM-DATE-TIME:2023-05-03T21:24:23.295678Z
#EXTINF:4.010667,
https://123.cloudfront.net/out/v1/index_1_34.ts
#EXT-X-PROGRAM-DATE-TIME:2023-05-03T21:24:27.306345Z
#EXTINF:4.010667,
https://123.cloudfront.net/out/v1/index_1_35.ts
#EXT-X-PROGRAM-DATE-TIME:2023-05-03T21:24:31.317012Z
#EXTINF:4.010667,
https://123.cloudfront.net/out/v1/index_1_36.ts
#EXT-X-PROGRAM-DATE-TIME:2023-05-03T21:24:35.327679Z
#EXTINF:4.010667,
https://123.cloudfront.net/out/v1/index_1_37.ts
#EXT-X-PROGRAM-DATE-TIME:2023-05-03T21:24:39.338346Z
#EXTINF:2.538667,
https://123.cloudfront.net/out/v1/index_1_38.ts
#EXT-X-DISCONTINUITY
#EXT-X-KEY:METHOD=NONE
#EXT-X-PROGRAM-DATE-TIME:2023-05-03T21:24:41.453Z
#EXTINF:2.0,
https://123.cloudfront.net/tm/asset_1080_4_8_00001.ts
#EXT-X-PROGRAM-DATE-TIME:2023-05-03T21:24:43.453Z
#EXTINF:2.0,
https://123.cloudfront.net/tm/asset_1080_4_8_00002.ts
#EXT-X-PROGRAM-DATE-TIME:2023-05-03T21:24:45.453Z
#EXTINF:2.0,
https://123.cloudfront.net/tm/asset_1080_4_8_00003.ts
```

In the client-side tracking JSON payload, the following values apply:

- `startTime`: `"PT18.581355S"`
- `startTimeInSeconds`: `18.581`
- `availProgramDateTime`: `"2023-05-03T21:24:41.453Z"`
- `adId`: `4603269`

Example : DASH

In the following example, the MediaTailor session gets a midroll in the manifest. Note that the `@start` attribute value of the second period, which is the ad period, has a value that's relative to the `MPD@availabilityStartTime` value. This value is the one that MediaTailor writes into the client-side tracking response `startTime` fields, for all sessions.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD availabilityStartTime="2022-10-05T19:38:39.263Z" minBufferTime="PT10S"
 minimumUpdatePeriod="PT2S" profiles="urn:mpeg:dash:profile:isoff-live:2011"
 publishTime="2023-05-03T22:06:48.411Z" suggestedPresentationDelay="PT10S"
 timeShiftBufferDepth="PT1M30S" type="dynamic" xmlns="urn:mpeg:dash:schema:mpd:2011"
 xmlns:scte35="urn:scte:scte35:2013:xml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/
ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd">
    <BaseURL>https://123.channel-assembly.mediatailor.us-west-2.amazonaws.com/v1/
channel/my-channel/</BaseURL>
    <Location>https://123.cloudfront.net/v1/
dash/94063eadf7d8c56e9e2edd84fdf897826a70d0df/MediaTailor-Live-HLS-DASH/channel/
channel1/dash.mpd?aws.sessionId=794a15e0-2a7f-4941-a537-9d71627984e5</Location>
    <Period id="1683151479166_1" start="PT5042H25M59.903S"
 xmlns="urn:mpeg:dash:schema:mpd:2011">
        <BaseURL>https://123.cloudfront.net/out/v1/f1a946be8efa45b0931ea35c9055fb74/
ddb73bf548a44551a0059c346226445a/eaa5485198bf497284559efb8172425e/</BaseURL>
        <AdaptationSet ...>
            ...
        </AdaptationSet>
    </Period>
    <Period id="1683151599194_1_1" start="PT5042H27M59.931S">
        <BaseURL>https://123.cloudfront.net/
tm/94063eadf7d8c56e9e2edd84fdf897826a70d0df/fpc5omz5wzd2rdepgieibp23ybyqyrme/</BaseURL>
        <AdaptationSet ...>
            ...
        </AdaptationSet>
    </Period>
</MPD>
```

In the client-side tracking JSON payload, the following values apply:

- `startTime`: `"PT5042H27M59.931S"`
- `startTimeInSeconds`: `18152879.931`
- `availProgramDateTime`: *`null`*

- `adId: 1683151599194_1_1`

Player controls and functionality for client-side ad tracking

MediaTailor client-side tracking metadata supports various player controls and functionality. The following list describes popular player controls.

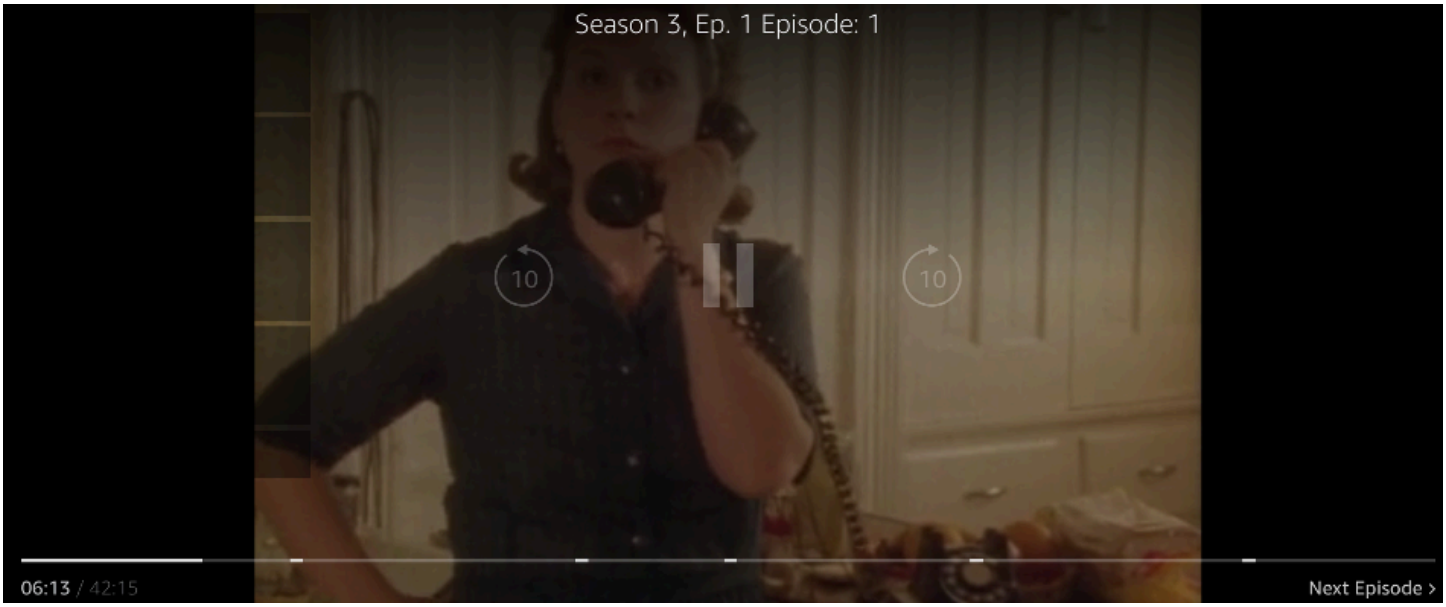
Topics

- Scrubbing
- Ad countdown timer
- Skippable ads
- Ad click-through
- Companion ads
- Interactive ads (SIMID)
- Interactive ads (VPAID)
- Icons for Google Why This Ad (WTA)

Scrubbing

To enhance the playback experience, the player can display ad positions in the playback timeline. MediaTailor makes these ad positions available in the form of `adStartTimeInSeconds` values in the client-side tracking response.

> **Note**
>
> Some streaming providers prevent scrubbing past an ad position.

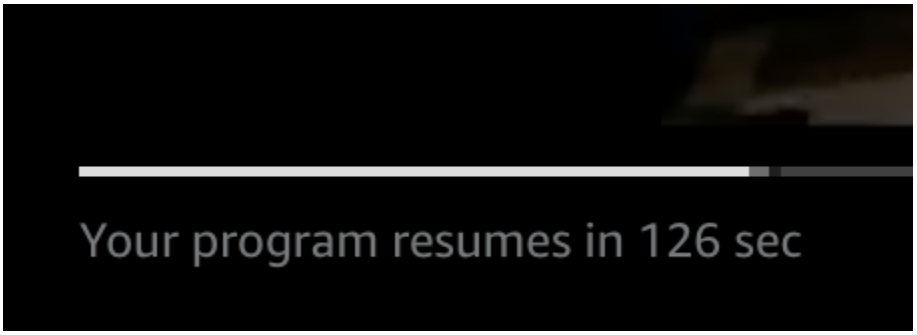


The following client-side tracking payload JSON response shows the avail (ad break) start time inside the root JSON object of the avails array. The player uses this data to show the location of the ad break on the player timeline, at 28 seconds.


```
{
  "avails": [
    {
      "adBreakTrackingEvents": [],
      "adMarkerDuration": null,
      "ads": [...],
      "availId": "7",
      "availProgramDateTime": null,
      "duration": "PT30S",
      "durationInSeconds": 30,
      "meta": null,
      "nonLinearAdsList": [],
      "startTime": "PT28S",
      "startTimeInSeconds": 28
    }
  ],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": "UFQxMk0zNC44NjhTXzIwMjMtMDctMjFUMjA6MjM6MDcuNzc1NzE2MzAyWl8x",
  "nonLinearAvails": []
}
```

Ad countdown timer

With MediaTailor you can use an ad countdown timer to help keep your audience engaged during ad-break viewing. The audience can use the timer to understand when the ad break ends and their program resumes.



The elements in the client-side tracking metadata that play a role in the ad countdown timer are `startTime`, `startTimeInSeconds`, `duration`, and `durationInSeconds`. The player uses this metadata, along with the session's elapsed time that it tracks separately, to determine when to display the timer and the value it should be counting down from.

The following client-side tracking payload JSON response shows the information needed to display an ad countdown timer.

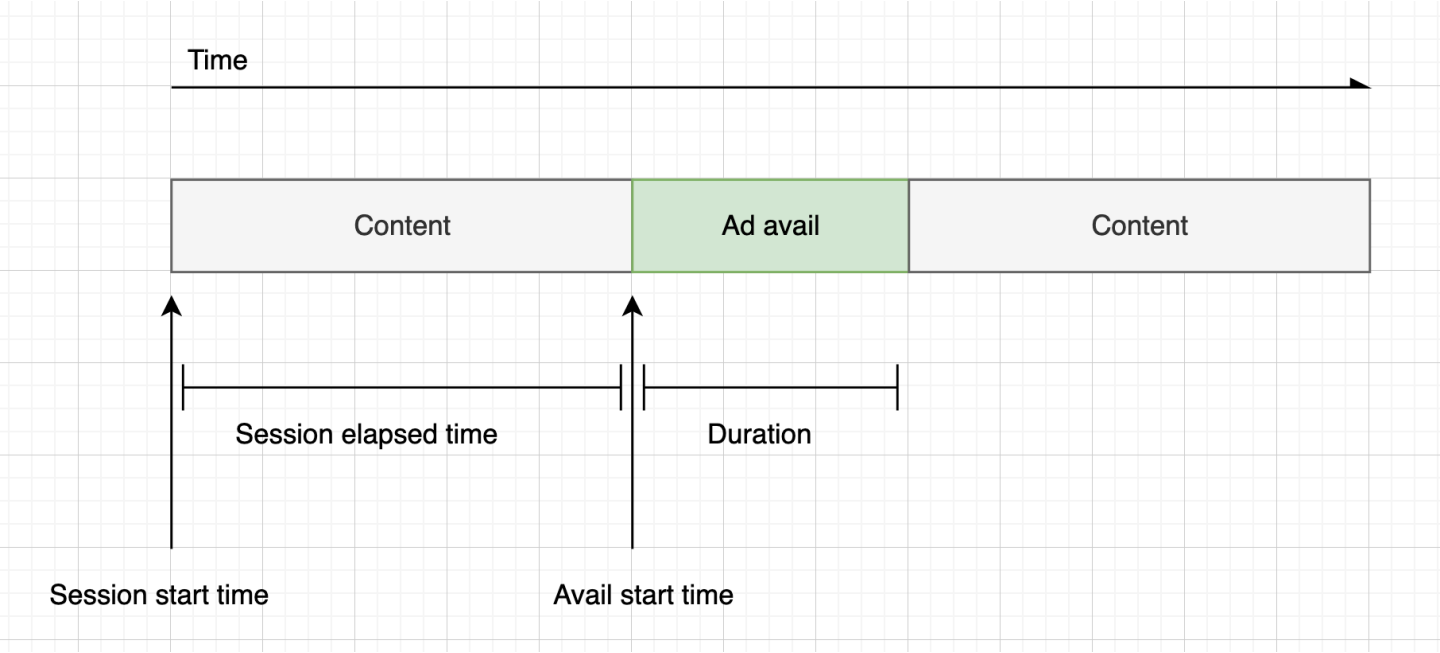

```
{
  "avails": [
    {
      "adBreakTrackingEvents": [],
      "adMarkerDuration": null,
      "ads": [...],
      "availId": "7",
      "availProgramDateTime": null,
      "duration": "PT30S",
      "durationInSeconds": 30,
      "meta": null,
      "nonLinearAdsList": [],
      "startTime": "PT28S",
      "startTimeInSeconds": 28
    }
  ],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": "UFQxMk0zNC44NjhTXzIwMjMtMDctMjFUMjA6MjM6MDcuNzc1NzE2MzAyWl8x",
  "nonLinearAvails": []
```

```
}
```

When the session's elapsed time reaches the avail's start time, the player displays a countdown timer with a value that matches the avail's duration. The countdown-timer value decreases as the elapsed time progresses beyond the avail's start time.

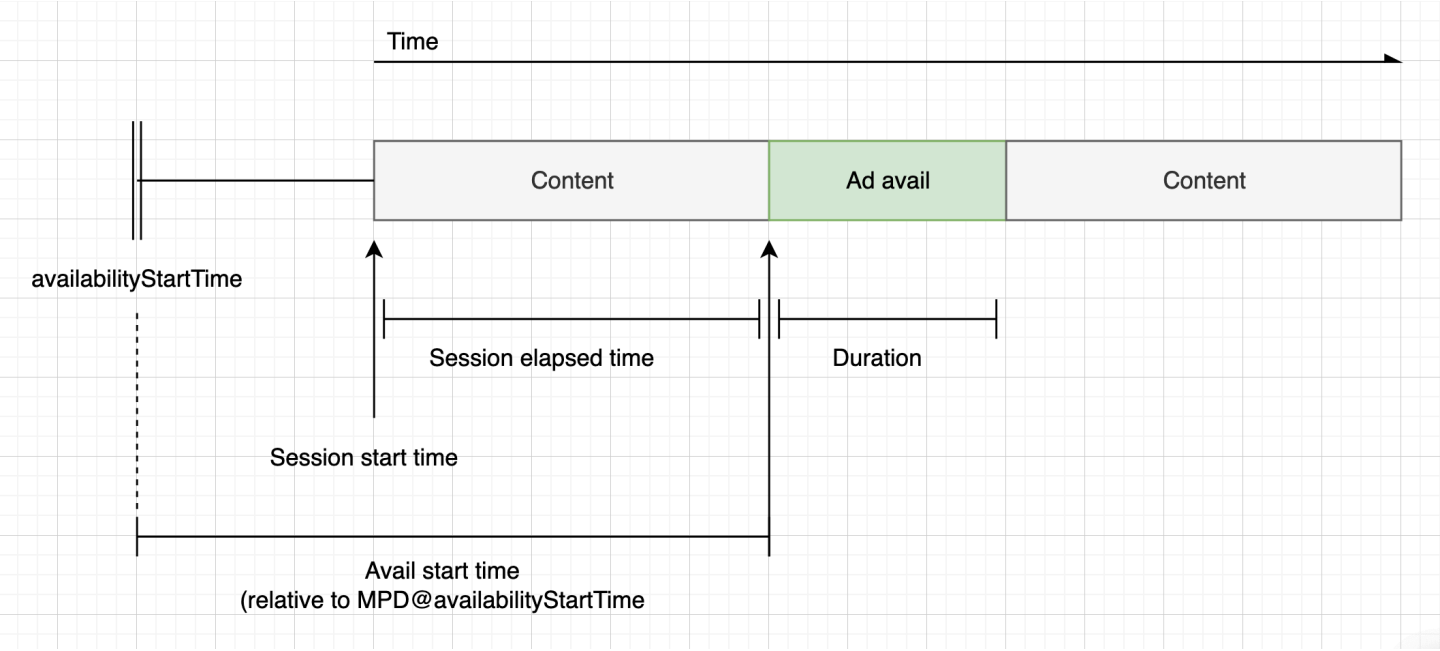
Example formula: Countdown timer for HLS (live and VOD) and DASH (VOD)

- `session_start_time` = the sum of all `EXT-INF` duration values - the duration value of the three newest `EXT-INF` media sequences
- timer value = `duration` - (`session_elapsed_time` - `startTime`)



Example formula: Countdown timer for DASH (live)

- `session_start_time` = (newest segment's `startTime` + `duration`) / `timescale` - `MPD@suggestedPresentationDelay`
- timer value = `duration` - (`session_elapsed_time` - `startTime`)



Skippable ads

Skippable ads are ad spots that allow the viewer to skip some of the ad to resume viewing the program. In VAST, the `Linear@skipOffset` attribute identifies a skippable ad.

The following VAST response shows how to use a skippable ad:

```
<?xml version="1.0" encoding="UTF-8"?>
<VAST xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="vast.xsd" version="3.0">
  <Ad>
    <InLine>
      ...
      <Creatives>
        <Creative id="1" sequence="1">
          <Linear skipoffset="00:00:05">
            <Duration>00:00:15</Duration>
            <MediaFiles>
              <MediaFile id="EMT" delivery="progressive" width="640" height="360"
 type="video/mp4" bitrate="143" scalable="true" maintainAspectRatio="true"><!
[CDATA[https://ads.com/file.mp4]]></MediaFile>
            </MediaFiles>
          </Linear>
        </Creative>
      </Creatives>
      ...
```

```
    </InLine>
  </Ad>
</VAST>
```

The following client-side tracking payload JSON response shows the ad metadata inside the `ads` array. The array contains the `skipOffset` value that MediaTailor obtained from the VAST response.

```
{
  "avails": [
    {
      "adBreakTrackingEvents": [],
      "adMarkerDuration": null,
      "ads": [
        {
          "adId": "1",
          "adParameters": "",
          "adProgramDateTime": "2023-07-31T16:11:40.693Z",
          "adSystem": "2.0",
          "adTitle": "AD-skiing-15",
          "adVerifications": [],
          "companionAds": [...],
          "creativeId": "1",
          "creativeSequence": "1",
          "duration": "PT15.015S",
          "durationInSeconds": 15.015,
          "extensions": [],
          "mediaFiles": {
            "mediaFilesList": [],
            "mezzanine": ""
          },
          "skipOffset": "00:00:05",
          "startTime": "PT9.943S",
          "startTimeInSeconds": 9.943,
          "trackingEvents": [
            {
              "beaconUrls": [
                "https://adserverbeaconing.com/v1/impression"
              ],
              "duration": "PT15.015S",
              "durationInSeconds": 15.015,
              "eventId": "2697726",
              "eventProgramDateTime": null,
```

```
              "eventType": "impression",
              "startTime": "PT9.943S",
              "startTimeInSeconds": 9.943
            }
          ],
          "vastAdId": ""
        }
      ],
      "availId": "2697726",
      "availProgramDateTime": "2023-07-31T16:11:40.693Z",
      "duration": "PT15.015S",
      "durationInSeconds": 15.015,
      "meta": null,
      "nonLinearAdsList": [],
      "startTime": "PT9.943S",
      "startTimeInSeconds": 9.943
    }
  ],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": "",
  "nonLinearAvails": []
}
```


Ad click-through

Click-through URIs allow advertisers to measure how successful an ad is in capturing viewers' attention. After a viewer clicks the active video frame of an ad in progress, a web browser opens the URI for the advertiser's home page or campaign landing page. The player developer determines the click behavior, such as overlaying a button or label on the ad video, with a message to click to learn more. Player developers often pause the ad's video after viewers click on the active video frame.

MediaTailor can parse and make available any linear video click-through event URLs returned in the VAST response. The following VAST response shows an ad click-through example.

```
<?xml version="1.0" encoding="UTF-8"?>
<VAST xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="vast.xsd" version="3.0">
  <Ad>
    <InLine>
      ...
      <Creatives>
        <Creative id="1" sequence="1">
          <Linear>
            <Duration>00:00:15</Duration>
            <MediaFiles>
```

```
              <MediaFile id="EMT" delivery="progressive" width="1280" height="720"
 type="video/mp4" bitrate="143" scalable="true" maintainAspectRatio="true"><!
[CDATA[https://ads.com/file.mp4]]></MediaFile>
            </MediaFiles>
            <VideoClicks>
              <ClickThrough id="EMT"><![CDATA[https://aws.amazon.com]]></ClickThrough>
              <ClickTracking id="EMT"><![CDATA[https://myads.com/beaconing/
event=clicktracking]]></ClickTracking>
            </VideoClicks>
          </Linear>
        </Creative>
      </Creatives>
      ...
    </InLine>
  </Ad>
</VAST>
```

The following client-side tracking payload JSON response shows how MediaTailor displays the click-through and click-tracking URLs inside the `trackingEvents` array. The `clickThrough` event type represents the click-through ad, and the `clickTracking` event type represents the click-tracking URL.

```
{
  "avails": [
    {
      "adBreakTrackingEvents": [],
      "adMarkerDuration": null,
      "ads": [
        {
          "adId": "1",
          "adParameters": "",
          "adProgramDateTime": "2023-07-31T16:53:40.577Z",
          "adSystem": "2.0",
          "adTitle": "1",
          "adVerifications": [],
          "companionAds": [],
          "creativeId": "00006",
          "creativeSequence": "1",
          "duration": "PT14.982S",
          "durationInSeconds": 14.982,
          "extensions": [],
          "mediaFiles": {
            "mediaFilesList": [],
```

```
            "mezzanine": ""
          },
          "skipOffset": null,
          "startTime": "PT39.339S",
          "startTimeInSeconds": 39.339,
          "trackingEvents": [
            {
              "beaconUrls": [
                "https://myads.com/beaconing/event=impression"
              ],
              "duration": "PT14.982S",
              "durationInSeconds": 14.982,
              "eventId": "2698188",
              "eventProgramDateTime": null,
              "eventType": "impression",
              "startTime": "PT39.339S",
              "startTimeInSeconds": 39.339
            },
            {
              "beaconUrls": [
                "https://aws.amazon.com"
              ],
              "duration": "PT14.982S",
              "durationInSeconds": 14.982,
              "eventId": "2698188",
              "eventProgramDateTime": null,
              "eventType": "clickThrough",
              "startTime": "PT39.339S",
              "startTimeInSeconds": 39.339
            },
            {
              "beaconUrls": [
                "https://myads.com/beaconing/event=clicktracking"
              ],
              "duration": "PT14.982S",
              "durationInSeconds": 14.982,
              "eventId": "2698795",
              "eventProgramDateTime": null,
              "eventType": "clickTracking",
              "startTime": "PT39.339S",
              "startTimeInSeconds": 39.339
            }
          ],
          "vastAdId": ""
```

```
        }
      ],
      "availId": "2698188",
      "availProgramDateTime": "2023-07-31T16:53:40.577Z",
      "duration": "PT14.982S",
      "durationInSeconds": 14.982,
      "meta": null,
      "nonLinearAdsList": [],
      "startTime": "PT39.339S",
      "startTimeInSeconds": 39.339
    }
  ],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": "UFQzOS4zMzlTXzIwMjMtMDctMzFUMTY6NTQ6MDQuODA1Mzk2NTI5Wl8x",
  "nonLinearAvails": []
}
```

Companion ads

A *companion ad* appears alongside a linear creative. Use companion ads to increase the effectiveness of an ad spot by displaying product, logo, and brand information. The display ad can feature Quick Response (QR) codes and clickable areas to promote audience engagement.

MediaTailor supports companion ads in the VAST response. It can pass through metadata from `StaticResource`, `iFrameResource`, and `HTMLResource` nodes, respectively.

The following VAST response shows an example location and format of the linear ad and companion ad.

```
<?xml version="1.0" encoding="UTF-8"?>
<VAST xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="vast.xsd" version="3.0">
  <Ad>
    <InLine>
      ...
      <Creatives>
        <Creative id="1" sequence="1">
          <Linear>
            <Duration>00:00:10</Duration>
            <MediaFiles>
```

```
              <MediaFile id="EMT" delivery="progressive" width="640" height="360"
 type="video/mp4" bitrate="143" scalable="true" maintainAspectRatio="true"><!
[CDATA[https://ads.com/file.mp4]]></MediaFile>
            </MediaFiles>
          </Linear>
        </Creative>
        <Creative id="2" sequence="1">
          <CompanionAds>
            <Companion id="2" width="300" height="250">
              <StaticResource creativeType="image/png"><![CDATA[https://emt.com/
companion/9973499273]]></StaticResource>
              <TrackingEvents>
                <Tracking event="creativeView"><![CDATA[https://beacon.com/1]]></
Tracking>
              </TrackingEvents>
              <CompanionClickThrough><![CDATA[https://beacon.com/2]]></
CompanionClickThrough>
            </Companion>
            <Companion id="3" width="728" height="90">
              <StaticResource creativeType="image/png"><![CDATA[https://emt.com/
companion/1238901823]]></StaticResource>
              <TrackingEvents>
                <Tracking event="creativeView"><![CDATA[https://beacon.com/3]]></
Tracking>
              </TrackingEvents>
              <CompanionClickThrough><![CDATA[https://beacon.com/4]]></
CompanionClickThrough>
            </Companion>
          </CcompanionAds>
        </Creative>
      </Creatives>
      ...
    </InLine>
  </Ad>
</VAST>
```

The data appears in the client-side tracking response in the `/avail/x/ads/y/companionAds` list. Each linear creative can contain up to 6 companion ads. As shown in the example below, the companion ads appear in a list

> **Note**
>
> As a best practice, application developers should implement logic to explicitly remove or unload the companion ad at the end of the creative.


```
{
  "avails": [
    {
      "adBreakTrackingEvents": [],
      "adMarkerDuration": null,
      "ads": [
        {
          "adId": "0",
          "adParameters": "",
          "adProgramDateTime": null,
          "adSystem": "EMT",
          "adTitle": "sample",
          "adVerifications": [],
          "companionAds": [
            {
              "adParameters": null,
              "altText": null,
              "attributes": {
                "adSlotId": null,
                "apiFramework": null,
                "assetHeight": null,
                "assetWidth": null,
                "expandedHeight": null,
                "expandedWidth": null,
                "height": "250",
                "id": "2",
                "pxratio": null,
                "renderingMode": null,
                "width": "300"
              },
              "companionClickThrough": "https://beacon.com/2",
              "companionClickTracking": null,
              "htmlResource": null,
              "iFrameResource": null,
              "sequence": "1",
              "staticResource": "https://emt.com/companion/9973499273",
```

```
            "trackingEvents": [
              {
                "beaconUrls": [
                  "https://beacon.com/1"
                ],
                "eventType": "creativeView"
              }
            ]
          },
          {
            "adParameters": null,
            "altText": null,
            "attributes": {
              "adSlotId": null,
              "apiFramework": null,
              "assetHeight": null,
              "assetWidth": null,
              "expandedHeight": null,
              "expandedWidth": null,
              "height": "90",
              "id": "3",
              "pxratio": null,
              "renderingMode": null,
              "width": "728"
            },
            "companionClickThrough": "https://beacon.com/4",
            "companionClickTracking": null,
            "htmlResource": null,
            "iFrameResource": null,
            "sequence": "1",
            "staticResource": "https://emt.com/companion/1238901823",
            "trackingEvents": [
              {
                "beaconUrls": [
                  "https://beacon.com/3"
                ],
                "eventType": "creativeView"
              }
            ]
          }
        ],
        "creativeId": "1",
        "creativeSequence": "1",
        "duration": "PT10S",
```

```
          "durationInSeconds": 10,
          "extensions": [],
          "mediaFiles": {
            "mediaFilesList": [],
            "mezzanine": ""
          },
          "skipOffset": null,
          "startTime": "PT0S",
          "startTimeInSeconds": 0,
          "trackingEvents": [
            {
              "beaconUrls": [
                "https://beacon.com/impression/1"
              ],
              "duration": "PT10S",
              "durationInSeconds": 10,
              "eventId": "0",
              "eventProgramDateTime": null,
              "eventType": "impression",
              "startTime": "PT0S",
              "startTimeInSeconds": 0
            }
          ],
          "vastAdId": ""
        }
      ],
      "availId": "0",
      "availProgramDateTime": null,
      "duration": "PT10S",
      "durationInSeconds": 10,
      "meta": null,
      "nonLinearAdsList": [],
      "startTime": "PT0S",
      "startTimeInSeconds": 0
    }
  ],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": "UFQxMFNfMjAyMy0wNy0wNlQyMToxMDowOC42NzQ4NDA1NjJaXzE%3D",
  "nonLinearAvails": []
}
```

Interactive ads (SIMID)

SecureInteractive Media Interface Definition (SIMID) is a standard for interactive advertising that was introduced in the VAST 4.x standard from the Interactive Advertising Bureau (IAB). SIMID decouples the loading of interactive elements from the primary linear creative on the player, referencing both in the VAST response. MediaTailor stitches in the primary creative to maintain the playback experience, and places metadata for the interactive components in the client-side tracking response.

In the following example VAST 4 response, the SIMID payload is inside the `InteractiveCreativeFile` node.

```
<?xml version="1.0"?>
<VAST xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="vast.xsd" version="3.0">
  <Ad id="1234567">
    <InLine>
      <AdSystem>SampleAdSystem</AdSystem>
      <AdTitle>Linear SIMID Example</AdTitle>
      <Description>SIMID example</Description>
      <Error>https://www.beacons.com/error</Error>
      <Impression>https://www.beacons.com/impression</Impression>
      <Creatives>
        <Creative sequence="1">
          <Linear>
            <Duration>00:00:15</Duration>
            <TrackingEvents>
                ...
            </TrackingEvents>
            <VideoClicks>
              <ClickThrough id="123">https://aws.amazon.com</ClickThrough>
              <ClickTracking id="123">https://www.beacons.com/click</ClickTracking>
            </VideoClicks>
            <MediaFiles>
              <MediaFile delivery="progressive" type="video/mp4">
                                https://interactive-ads.com/interactive-media-ad-
sample/media/file.mp4
                            </MediaFile>
              <InteractiveCreativeFile type="text/html" apiFramework="SIMID"
 variableDuration="true">
                                https://interactive-ads.com/interactive-media-ad-
sample/sample_simid.html
                            </InteractiveCreativeFile>
```

```
            </MediaFiles>
          </Linear>
        </Creative>
      </Creatives>
    </InLine>
  </Ad>
</VAST>
```

In the following VAST 3 response, the SIMID payload is inside the `Extensions` node.

```
<?xml version="1.0"?>
<VAST xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="vast.xsd" version="3.0">
  <Ad id="1234567">
    <InLine>
      <AdSystem>SampleAdSystem</AdSystem>
      <AdTitle>Linear SIMID Example</AdTitle>
      <Description>SIMID example</Description>
      <Impression>https://www.beacons.com/impression</Impression>
      <Creatives>
        <Creative id="1" sequence="1">
          <Linear>
            <Duration>00:00:15</Duration>
            <TrackingEvents>
                ...
            </TrackingEvents>
            <VideoClicks>
              <ClickThrough id="123">https://aws.amazon.com</ClickThrough>
              <ClickTracking id="123">https://myads.com/beaconing/event=clicktracking</
ClickTracking>
            </VideoClicks>
            <MediaFiles>
              <MediaFile delivery="progressive" type="video/mp4">
                                https://interactive-ads.com/interactive-media-ad-
sample/media/file.mp4
                            </MediaFile>
            </MediaFiles>
          </Linear>
        </Creative>
      </Creatives>
      <Extensions>
        <Extension type="InteractiveCreativeFile">
```

```
          <InteractiveCreativeFile type="text/html" apiFramework="SIMID"
 variableDuration="true">
            https://interactive-ads.com/interactive-media-ad-sample/sample_simid.html
          </InteractiveCreativeFile>
        </Extension>
      </Extensions>
    </InLine>
  </Ad>
</VAST>
```

In the following client-side tracking response, the SIMID data appears in the `/avails/x/ads/y/extensions` list.

```
{
  "avails": [
    {
      "adBreakTrackingEvents": [],
      "adMarkerDuration": null,
      "ads": [
        {
          "adId": "1",
          "adParameters": "",
          "adProgramDateTime": "2023-07-31T16:53:40.577Z",
          "adSystem": "2.0",
          "adTitle": "Linear SIMID Example",
          "adVerifications": [],
          "companionAds": [],
          "creativeId": "1",
          "creativeSequence": "1",
          "duration": "PT14.982S",
          "durationInSeconds": 14.982,
          "extensions": [
            {
              "content": "<InteractiveCreativeFile type=\"text/html\" apiFramework=
\"SIMID\" variableDuration=\"true\">\nhttps://interactive-ads.com/interactive-media-ad-
sample/sample_simid.html</InteractiveCreativeFile>",
              "type": "InteractiveCreativeFile"
            }
          ],
          "mediaFiles": {
            "mediaFilesList": [],
            "mezzanine": ""
          },
```

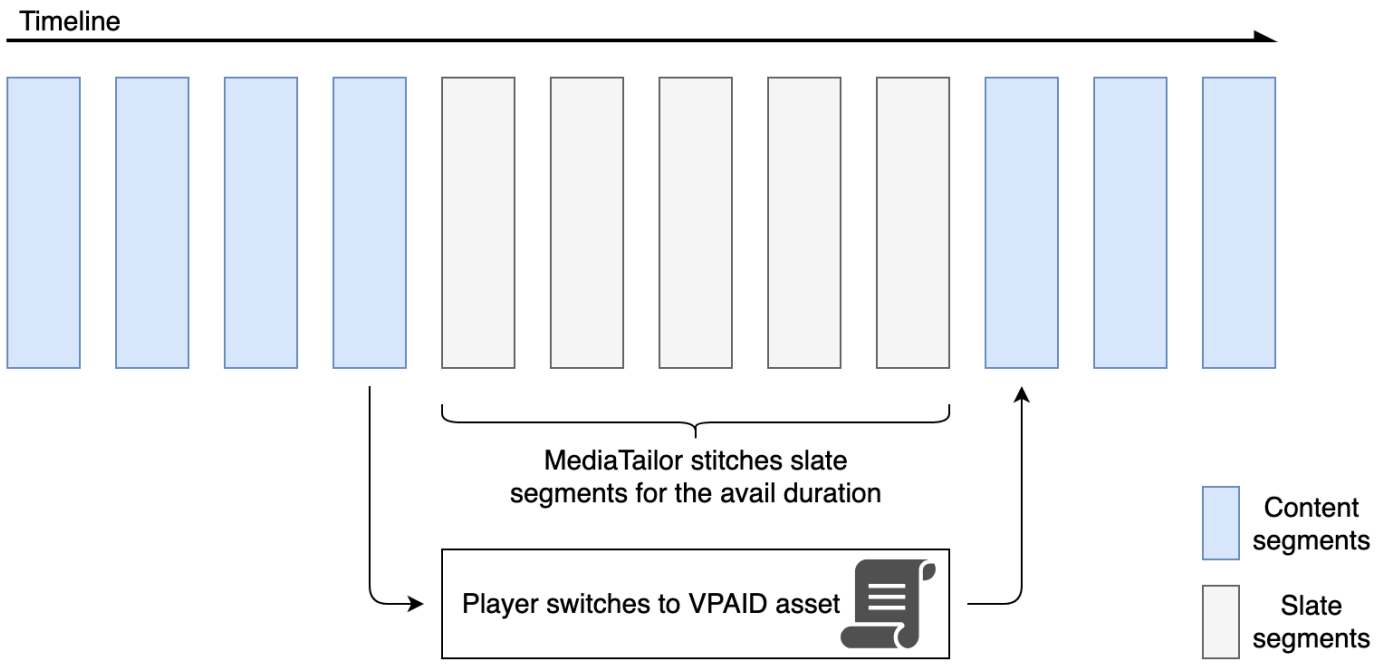
```
          "skipOffset": null,
          "startTime": "PT39.339S",
          "startTimeInSeconds": 39.339,
          "trackingEvents": [
            {
              "beaconUrls": [
                "https://myads.com/beaconing/event=impression"
              ],
              "duration": "PT14.982S",
              "durationInSeconds": 14.982,
              "eventId": "2698188",
              "eventProgramDateTime": null,
              "eventType": "impression",
              "startTime": "PT39.339S",
              "startTimeInSeconds": 39.339
            },
            {
              "beaconUrls": [
                "https://aws.amazon.com"
              ],
              "duration": "PT14.982S",
              "durationInSeconds": 14.982,
              "eventId": "2698188",
              "eventProgramDateTime": null,
              "eventType": "clickThrough",
              "startTime": "PT39.339S",
              "startTimeInSeconds": 39.339
            },
            {
              "beaconUrls": [
                "https://myads.com/beaconing/event=clicktracking"
              ],
              "duration": "PT14.982S",
              "durationInSeconds": 14.982,
              "eventId": "2698795",
              "eventProgramDateTime": null,
              "eventType": "clickTracking",
              "startTime": "PT39.339S",
              "startTimeInSeconds": 39.339
            }
          ],
          "vastAdId": ""
        }
      ],
```

```
      "availId": "2698188",
      "availProgramDateTime": "2023-07-31T16:53:40.577Z",
      "duration": "PT14.982S",
      "durationInSeconds": 14.982,
      "meta": null,
      "nonLinearAdsList": [],
      "startTime": "PT39.339S",
      "startTimeInSeconds": 39.339
    }
  ],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": "UFQzOS4zMzlTXzIwMjMtMDctMzFUMTY6NTQ6MDQuODA1Mzk2NTI5Wl8x",
  "nonLinearAvails": []
}
```


Interactive ads (VPAID)

The *Video Player Ad Interface Definition* (VPAID) specifies the protocol between the ad and the video player that enables ad interactivity and other functionality. For live streams, MediaTailor supports the VPAID format by stitching slate segments in for the duration of the avail, and placing metadata for the VPAID creatives in the client-side tracking response that the video player consumes. The player downloads the VPAID files and plays the linear creative and executes the client's scripts. The player should *not* ever play the slate segments.

> **Note**
>
> VPAID is deprecated as of VAST 4.1.



The following example shows the VPAID content in the VAST response.

```
<?xml version="1.0"?>
<VAST xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="vast.xsd" version="3.0">
  <Ad id="1234567">
    <InLine>
      <AdSystem>GDFP</AdSystem>
      <AdTitle>VPAID</AdTitle>
      <Description>Vpaid Linear Video Ad</Description>
      <Error>http://www.example.com/error</Error>
      <Impression>http://www.example.com/impression</Impression>
      <Creatives>
        <Creative sequence="1">
          <Linear>
            <Duration>00:00:00</Duration>
            <TrackingEvents>
              <Tracking event="start">http://www.example.com/start</Tracking>
              <Tracking event="firstQuartile">http://www.example.com/firstQuartile</
Tracking>
              <Tracking event="midpoint">http://www.example.com/midpoint</Tracking>
              <Tracking event="thirdQuartile">http://www.example.com/thirdQuartile</
Tracking>
```

```
            <Tracking event="complete">http://www.example.com/complete</Tracking>
            <Tracking event="mute">http://www.example.com/mute</Tracking>
            <Tracking event="unmute">http://www.example.com/unmute</Tracking>
            <Tracking event="rewind">http://www.example.com/rewind</Tracking>
            <Tracking event="pause">http://www.example.com/pause</Tracking>
            <Tracking event="resume">http://www.example.com/resume</Tracking>
            <Tracking event="fullscreen">http://www.example.com/fullscreen</Tracking>
            <Tracking event="creativeView">http://www.example.com/creativeView</
Tracking>
            <Tracking event="acceptInvitation">http://www.example.com/
acceptInvitation</Tracking>
          </TrackingEvents>
          <AdParameters><![CDATA[ {"videos":[ {"url":"https://my-ads.com/interactive-
media-ads/media/media_linear_VPAID.mp4","mimetype":"video/mp4"}]} ]]></AdParameters>
          <VideoClicks>
            <ClickThrough id="123">http://google.com</ClickThrough>
            <ClickTracking id="123">http://www.example.com/click</ClickTracking>
          </VideoClicks>
          <MediaFiles>
            <MediaFile delivery="progressive" apiFramework="VPAID" type="application/
javascript" width="640" height="480"> https://googleads.github.io/googleads-ima-html5/
vpaid/linear/VpaidVideoAd.js </MediaFile>
          </MediaFiles>
        </Linear>
      </Creative>
    </Creatives>
  </InLine>
 </Ad>
</VAST>
```

The following example shows the tracking information.

```
{
  "avails": [
    {
      "adBreakTrackingEvents": [],
      "adMarkerDuration": null,
      "ads": [
        {
          "adId": "1",
          "adParameters": "",
          "adProgramDateTime": "2023-07-31T16:53:40.577Z",
          "adSystem": "2.0",
```

```
          "adTitle": "1",
          "adVerifications": [],
          "companionAds": [],
          "creativeId": "00006",
          "creativeSequence": "1",
          "duration": "PT14.982S",
          "durationInSeconds": 14.982,
          "extensions": [],
          "mediaFiles": {
            "mediaFilesList": [],
            "mezzanine": ""
          },
          "skipOffset": null,
          "startTime": "PT39.339S",
          "startTimeInSeconds": 39.339,
          "trackingEvents": [
            {
              "beaconUrls": [
                "https://myads.com/beaconing/event=impression"
              ],
              "duration": "PT14.982S",
              "durationInSeconds": 14.982,
              "eventId": "2698188",
              "eventProgramDateTime": null,
              "eventType": "impression",
              "startTime": "PT39.339S",
              "startTimeInSeconds": 39.339
            },
            {
              "beaconUrls": [
                "https://aws.amazon.com"
              ],
              "duration": "PT14.982S",
              "durationInSeconds": 14.982,
              "eventId": "2698188",
              "eventProgramDateTime": null,
              "eventType": "clickThrough",
              "startTime": "PT39.339S",
              "startTimeInSeconds": 39.339
            },
            {
              "beaconUrls": [
                "https://myads.com/beaconing/event=clicktracking"
              ],
```

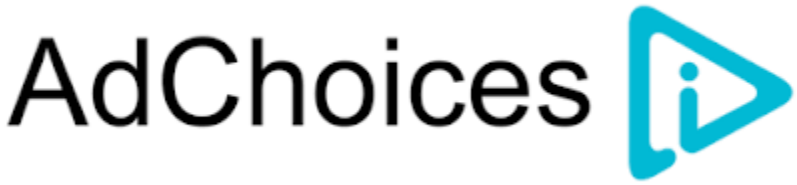
```
              "duration": "PT14.982S",
              "durationInSeconds": 14.982,
              "eventId": "2698795",
              "eventProgramDateTime": null,
              "eventType": "clickTracking",
              "startTime": "PT39.339S",
              "startTimeInSeconds": 39.339
            }
          ],
          "vastAdId": ""
        }
      ],
      "availId": "2698188",
      "availProgramDateTime": "2023-07-31T16:53:40.577Z",
      "duration": "PT14.982S",
      "durationInSeconds": 14.982,
      "meta": null,
      "nonLinearAdsList": [],
      "startTime": "PT39.339S",
      "startTimeInSeconds": 39.339
    }
  ],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": "UFQzOS4zMzlTXzIwMjMtMDctMzFUMTY6NTQ6MDQuODA1Mzk2NTI5Wl8x",
  "nonLinearAvails": []
}{
  "avails": [
    {
      "adBreakTrackingEvents": [],
      "adMarkerDuration": null,
      "ads": [
        {
          "adId": "2922274",
          "adParameters": "",
          "adProgramDateTime": "2023-08-14T19:49:53.998Z",
          "adSystem": "Innovid Ads",
          "adTitle": "VPAID",
          "adVerifications": [],
          "companionAds": [],
          "creativeId": "",
          "creativeSequence": "",
          "duration": "PT16.016S",
          "durationInSeconds": 16.016,
```

```
          "extensions": [],
          "mediaFiles": {
            "mediaFilesList": [
              {
                "apiFramework": "VPAID",
                "bitrate": 0,
                "codec": null,
                "delivery": "progressive",
                "height": 9,
                "id": "",
                "maintainAspectRatio": false,
                "maxBitrate": 0,
                "mediaFileUri": "http://my-ads.com/mobileapps/js/vpaid/1h41kg?
cb=178344c0-8e67-281a-58ca-962e4987cd60&deviceid=&ivc=",
                "mediaType": "application/javascript",
                "minBitrate": 0,
                "scalable": false,
                "width": 16
              }
            ],
            "mezzanine": "http://my-ads.com/mobileapps/js/vpaid/1h41kg?
cb=178344c0-8e67-281a-58ca-962e4987cd60&deviceid=&ivc="
          },
          "skipOffset": null,
          "startTime": "PT8M42.289S",
          "startTimeInSeconds": 522.289,
          "trackingEvents": [
            {
              "beaconUrls": [
                "about:blank"
              ],
              "duration": "PT16.016S",
              "durationInSeconds": 16.016,
              "eventId": "2922274",
              "eventProgramDateTime": null,
              "eventType": "impression",
              "startTime": "PT8M42.289S",
              "startTimeInSeconds": 522.289
            }
          ],
          "vastAdId": "1h41kg"
        }
      ],
      "availId": "2922274",
```

```
      "availProgramDateTime": "2023-08-14T19:49:53.998Z",
      "duration": "PT16.016S",
      "durationInSeconds": 16.016,
      "meta": null,
      "nonLinearAdsList": [],
      "startTime": "PT8M42.289S",
      "startTimeInSeconds": 522.289
    }
  ],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": "UFQ4TTQyLjI4OVNfMjAyMy0wOC0xNFQxOTo1MDo0MS4zOTc5MjAzODVaXzE%3D",
  "nonLinearAvails": []
}
```


Icons for Google Why This Ad (WTA)

AdChoices is an industry standard that provides viewers with information about the ads they see, including how those ads were targeted to them.



The MediaTailor client-side tracking API supports icon metadata carried in the VAST extensions node of the VAST response. For more information about WTA in the VAST response, see this sample VAST XML response.

> **Note**
>
> MediaTailor currently supports VAST version 3 only.

```
<VAST>
    <Ad>
    <InLine>
       ...
      <Extensions>
        <Extension type="IconClickFallbackImages">
```

```
          <IconClickFallbackImages program="GoogleWhyThisAd">
            <IconClickFallbackImage width="400" height="150">
              <AltText>Alt icon fallback</AltText>
              <StaticResource creativeType="image/png"><![CDATA[https://
storage.googleapis.com/interactive-media-ads/images/wta_dialog.png]]></StaticResource>
            </IconClickFallbackImage>
          </IconClickFallbackImages>
          <IconClickFallbackImages program="AdChoices">
            <IconClickFallbackImage width="400" height="150">
              <AltText>Alt icon fallback</AltText>
              <StaticResource creativeType="image/png"><![CDATA[https://
storage.googleapis.com/interactive-media-ads/images/wta_dialog.png?size=1x]]></
StaticResource>
            </IconClickFallbackImage>
            <IconClickFallbackImage width="800" height="300">
              <AltText>Alt icon fallback</AltText>
              <StaticResource creativeType="image/png"><![CDATA[https://
storage.googleapis.com/interactive-media-ads/images/wta_dialog.png?size=2x]]></
StaticResource>
            </IconClickFallbackImage>
          </IconClickFallbackImages>
        </Extension>
      </Extensions>
    </InLine>
  </Ad>
</VAST>
```

The following example shows the client-side tracking response in the `/avails/x/ads/y/extensions` list.

```
{
  "avails": [
    {
      "adBreakTrackingEvents": [],
      "adMarkerDuration": null,
      "ads": [
        {
          "adId": "0",
          "adParameters": "",
          "adProgramDateTime": null,
          "adSystem": "GDFP",
          "adTitle": "Google Why This Ad VAST 3 Sample",
          "adVerifications": [],
```

```
          "companionAds": [],
          "creativeId": "7891011",
          "creativeSequence": "1",
          "duration": "PT10S",
          "durationInSeconds": 10,
          "extensions": [
            {
              "content": "<IconClickFallbackImages program=\"GoogleWhyThisAd\">
                          <IconClickFallbackImage height=\"150\" width=\"400\">
                          <AltText>Alt icon fallback</AltText>
                          <StaticResource creativeType=\"image/png\"><![CDATA[https://
storage.googleapis.com/interactive-media-ads/images/wta_dialog.png]]>
                          </StaticResource>
                          </IconClickFallbackImage>
                          </IconClickFallbackImages>
                          <IconClickFallbackImages program=\"AdChoices\">
                          <IconClickFallbackImage height=\"150\" width=\"400\">
                          <AltText>Alt icon fallback</AltText>
                          <StaticResource creativeType=\"image/png\"><![CDATA[https://
storage.googleapis.com/interactive-media-ads/images/wta_dialog.png?size=1x]]>
                          </StaticResource>
                          </IconClickFallbackImage>
                          <IconClickFallbackImage height=\"300\" width=\"800\">
                          <AltText>Alt icon fallback</AltText>
                          <StaticResource creativeType=\"image/png\"><![CDATA[https://
storage.googleapis.com/interactive-media-ads/images/wta_dialog.png?size=2x]]>
                          </StaticResource>
                          </IconClickFallbackImage>
                          </IconClickFallbackImages>",
              "type": "IconClickFallbackImages"
            }
          ],
          "mediaFiles": {
            "mediaFilesList": [],
            "mezzanine": ""
          },
          "skipOffset": "00:00:03",
          "startTime": "PT0S",
          "startTimeInSeconds": 0,
          "trackingEvents": [
            {
              "beaconUrls": [
                "https://example.com/view"
              ],
```

```
              "duration": "PT10S",
              "durationInSeconds": 10,
              "eventId": "0",
              "eventProgramDateTime": null,
              "eventType": "impression",
              "startTime": "PT0S",
              "startTimeInSeconds": 0
            }
          ],
          "vastAdId": "123456"
        }
      ],
      "availId": "0",
      "availProgramDateTime": null,
      "duration": "PT10S",
      "durationInSeconds": 10,
      "meta": null,
      "nonLinearAdsList": [],
      "startTime": "PT0S",
      "startTimeInSeconds": 0
    }
  ],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": "UFQxMFNfMjAyMy0wNy0wNlQyMDo0MToxNy45NDE4MDM0NDhaXzE%3D",
  "nonLinearAvails": []
}
```


Client-side beaconing

With the client-side tracking `startTimeInSeconds` element, you can use MediaTailor to support beacons timing.

The following JSON response shows the main beacon types: impressions, start, quartiles, and completion.

> **Note**
>
> The Interactive Advertising Bureau (IAB) Video Impression Measurement Guidelines state that an impression requires the ad content to load client-side and, at a minimum, begin time to render into the player. For more information, see Digital Video Ad Serving Template (VAST) on the IAB website.

```
{
  "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,
      "startTime": "PT17.817798612S",
      "startTimeInSeconds": 17.817
    }
  ]
```

```
}
```

Hybrid mode with server-side ad beacons

MediaTailor supports a hybrid mode for session tracking. In this mode, the service emits playback-related ad-tracking events, but makes the complete client-side tracking payload available for the session

To enable hybrid tracking using playback prefixes, from the player initialize a new MediaTailor playback session using a request in one of the following formats, according to your protocol:

Example : HLS format

```
POST master.m3u8
    {
        "adsParams": {
           "deviceType": "ipad"
       },
       "reportingMode":"server"
    }
```

Example : DASH format

```
POST manifest.mpd
    {
        "adsParams": {
           "deviceType": "ipad"
       },
       "reportingMode":"server"
    }
```

MediaTailor maintains the following tracking events in hybrid mode:

- Impression
- Start
- First quartile
- Midpoint
- Third quartile
- Complete
- `breakStart` (vmap)

- `breakEnd` (vmap)

Client-side ad-tracking integrations

This section describes integrations between MediaTailor and various client-side ad-tracking servers.

Topics

- Open Measurement SDK
- Datazoom free player SDKs
- Roku Advertising Framework (RAF)
- TheoPlayer
- MediaTailor SDK

Open Measurement SDK

The Interactive Advertising Bureau (IAB) Open Measurement SDK (OM SDK) facilitates third-party viewability and verification measurement for ads served to web-video and native-app environments.

For older VAST version 3 documents, verification code should be loaded with the Extension node, with extension type `AdVerifications`. The root of the extension node is an `AdVerifications` node with the same schema as the VAST 4.1 element.

To facilitate easier adoption of the OM SDK, MediaTailor has partnered with Datazoom to provide free player SDKs that are configured and verified for Open Measurement. For more information, see Datazoom free player SDKs.

> **Note**
>
> MediaTailor currently supports VAST version 3 only.

Example : Verification node in VAST 3, prior to Version 4.1

```
...
<Extensions>
    <Extension type="AdVerifications">
        <AdVerifications>
```

```
            <Verification vendor="company.com-omid">
                <JavaScriptResource apiFramework="omid" browserOptional="true">
                    <![CDATA[https://verification.com/omid_verification.js]]>
                </JavaScriptResource>
                <TrackingEvents>
                    <Tracking event="verificationNotExecuted">
                        <![CDATA[https://verification.com/trackingurl]]>
                    </Tracking>
                </TrackingEvents>
                <VerificationParameters>
                    <![CDATA[verification params key/value pairs]]>
                </VerificationParameters>
            </Verification>
        </AdVerifications>
    </Extension>
</Extensions>
```

MediaTailor extracts the `AdVerifications` data from the `<Extensions>` node and places it into the `adVerifications` array in the client-side tracking response.

Example : adVerifications array in client-side tracking response

```
{
  "avails": [
    {
      "adBreakTrackingEvents": [],
      "adMarkerDuration": null,
      "ads": [
        {
          "adId": "3062770",
          "adParameters": "",
          "adProgramDateTime": "2023-08-23T16:25:40.914Z",
          "adSystem": "2.0",
          "adTitle": "AD-polarbear-15",
          "adVerifications": [
            {
              "executableResource": [],
              "javaScriptResource": [
                {
                  "apiFramework": "omid",
                  "browserOptional": "true",
                  "uri": "https://verification.com/omid_verification.js"
                }
```

```
            ],
            "trackingEvents": [
              {
                "event": "verificationNotExecuted",
                "uri": "https://verification.com/trackingurl"
              }
            ],
            "vendor": "company.com-omid",
            "verificationParameters": "verification params key value pairs"
          }
        ],
        "companionAds": [],
        "creativeId": "00006",
        "creativeSequence": "1",
        "duration": "PT14.982S",
        "durationInSeconds": 14.982,
        "extensions": [
          {
            "content": "<AdVerifications>\n\t\t\t\t\t\t<Verification vendor=
\"company.com-omid\">\n\t\t\t\t\t\t\t<JavaScriptResource apiFramework=\"omid\"
 browserOptional=\"true\"><![CDATA[https://verification.com/omid_verification.js;]]></
JavaScriptResource>\n\t\t\t\t\t\t\t<TrackingEvents>\n\t\t\t\t\t\t\t\t<Tracking event=
\"verificationNotExecuted\"><![CDATA[;https://verification.com/trackingurl;]]></
Tracking>\n\t\t\t\t\t\t\t</TrackingEvents>\n\t\t\t\t\t\t\t<VerificationParameters><!
[CDATA[verification params key/value pairs;]]></VerificationParameters>\n\t\t\t\t\t</
Verification>\n\t\t\t\t</AdVerifications>",
            "type": "AdVerifications"
          }
        ],
        "mediaFiles": {
          "mediaFilesList": [],
          "mezzanine": ""
        },
        "skipOffset": null,
        "startTime": "PT10.11S",
        "startTimeInSeconds": 10.11,
        "trackingEvents": [
          {
            "beaconUrls": [
              "https://n8ljfs0h09.execute-api.us-west-2.amazonaws.com/v1/impression"
            ],
            "duration": "PT14.982S",
            "durationInSeconds": 14.982,
            "eventId": "3062770",
```

```
              "eventProgramDateTime": null,
              "eventType": "impression",
              "startTime": "PT10.11S",
              "startTimeInSeconds": 10.11
            }
          ],
          "vastAdId": ""
        }
      ],
      "availId": "3062770",
      "availProgramDateTime": "2023-08-23T16:25:40.914Z",
      "duration": "PT14.982S",
      "durationInSeconds": 14.982,
      "meta": null,
      "nonLinearAdsList": [],
      "startTime": "PT10.11S",
      "startTimeInSeconds": 10.11
    }
  ],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": "UFQxMC4xMVNfMjAyMy0wOC0yM1QxNjoyNjoyNC4yNDYxMDIxOTBaXzE%3D",
  "nonLinearAvails": []
}
```


> **Note**
>
> Engage with the IAB Tech Lab to ensure that applications are certified annually to ensure compliance.

For more information about the OM SDK, see Open Measurement SDK on the IAB Tech Lab website.

Datazoom free player SDKs

To facilitate easier adoption of the player SDKs, MediaTailor has partnered with Datazoom to provide free player SDKs that are configured and tested with the Client-side AWS Elemental MediaTailor integration with Google Ad Manager and the IAB Tech Open Measurement SDK.

The Datazoom player SDK supports these features:

- Live and VOD playlists
- DASH and HLS specifications
- Player vendor support for Bitmovin, exoplayer, Android media player, Apple AVPlayer, Brightcove, Chromecast Receiver, Dash.js, hls.js, JWPlayer, Shaka player, THEO player, Video.js, Roku and more
- IAB Tech Lab Open Measurement certification, where available on selected devices
- Click-through event handling
- Ad-event dispatchers, such as ad count down timers, ad overlay and non-linear events, ad-break start, ad-break end
- Client-side ad beaconing
- Google Programmatic Access Library (PAL) SDK, as an optional configuration setting

Datazoom also offers a paid analytics and telemetry service that the player SDKs support. Customers can opt into and control player SDK telemetry from the Datazoom management console. To access the Datazoom player SDKs and to learn more about the value-added telemetry and analytics service, use the contact information on the Datazoom site.

Roku Advertising Framework (RAF)

The Roku Ad Framework (RAF) maintains a consistent ad experience across the Roku platform. All channels, including video advertisements, must meet Roku's certification requirements for RAF. Notably, the app must always use client-side event firing through RAF. MediaTailor, as a server-side ad insertion (SSAI) provider, supports client-side event firing. The RAFX SSAI Adapters provide interfaces to both SSAI manifest servers, or stitchers, and RAF. These interfaces include:

- Parsing the `masterURL` response and extracting `playURL`, `AdURL`, and ad metadata.
- Transforming MediaTailor SSAI ad metadata into RAF-usable ad metadata, and configuring RAF for playback.
- Observing stream events and timed metadata.
- Matching the stream events, ad metadata, and firing-event pixels on time.
- Pinging/polling the `AdURL`, as required by the MediaTailor SSAI manifest server, then parsing and re-configuring RAF.

For more information about SSAI adapters for RAF, see Implementing Server-Side Ad Insertion Using Roku Adapters on the Roku website.

TheoPlayer

TheoPlayer integration with MediaTailor does the following:

- Provides functionality to support MediaTailor client-side event tracking for HLS and DASH for both VOD and live workflows.
- Supports sending tracking beacons only for linear ads.
- Disables seeking during an ad. However, no logic is in place for playing an ad when the user seeks past the ad break.

For more information about SSAI in TheoPlayer, and to review the web, Android, iOS, and tvOS SDKs for MediaTailor, see MediaTailor on the TheoPlayer website.

MediaTailor SDK

AWS Elemental maintains a JavaScript-based software-development kit (SDK). AWS Elemental provides the SDK as-is, with no implied warranty. Use the SDK as a reference demonstration to streamline your onboarding to using MediaTailor. The SDK shows how to interact with the MediaTailor client-side tracking API. The SDK implements client-side ad tracking and reporting for HTML5-based players. The SDK initializes a MediaTailor client-side reporting session, then periodically requests ad-tracking information. During playback, the SDK emits ad tracking events when new ad events are detected.

The MediaTailor SDK supports these features:

- Live and VOD playlists
- DASH and HLS specifications
- Click-through event handling
- Ad-event dispatchers
- Custom event hooks
- Client-side ad beaconing. For more information about sending ad beacons, see Client-side beaconing.

> **Note**
>
> Submit an AWS Support ticket to receive a sample JavaScript SDK for MediaTailor. You'll receive a download link for the package and its files.

Paging through ad beacons with GetTracking

Use the `GetTracking` endpoint to narrow the number of ad returned to a player. For instance, if a manifest window is wide, spanning a lot of time, the number of returned ad beacons can impact player performance.

`GetTracking` returns a `NextToken` value you can use to narrow the number of returned beacons by paging through the list of returned beacons. You can cycle through the `NextToken` values to find the desired value of an ad beacon's `StartTimeInSeconds` field.

- On the first call to `GetTracking`, all possible ads falling in the manifest window are returned, including a `NextToken` and value of each.
- If a `GetTracking` request does *not* include a `NextToken`, all ads in the manifest window are returned.
- If a `GetTracking` request contains a `NextToken` but there are no new beacons to return, MediaTailor returns the same value for `NextToken` that you sent on the original request.
- When there are no more beacons corresponding to an ad, `GetTracking` removes the ad from its response.
- Tokens from `GetTracking` expire after 24 hours. If a `NextToken` value is greater than 24 hours old, the next call to `GetTracking` returns a null-value `NextToken`.

Generalized calling sequence of GetTracking from player

From the client player, a `GetTracking` request is a POST with a request body that contains the `NextToken` and ads and beacons related to the token.

```
https://YouMediaTailorUrl/v1/tracking
{

     "NextToken": "value"
     .
```

```
      .
      .
}
```

The general sequence for using `GetTracking` with `NextToken` is as follows:

1. Make the first call to `GetTracking`.

   All ads and beacons and the first `NextToken` for subsequent calls are returned.
2. If the value of `NextToken` is null, MediaTailor returns all ad beacons.
3. If the `NextToken` is expired, MediaTailor returns an HTTP return code 400 error message.

   Make a fresh call to `GetTracking` to retrieve valid `NextToken`s.
4. Scan the entire response to find the `StartTimeInSeconds` of an ad beacon that is in the desired range.
5. Make a new call to `GetTracking` with the value of `NextToken` associated with the desired `StartTimeInSeconds`.
6. If necessary, cycle again through the returned ads until you find the exact ones you want to play.

Extended example

This example shows how to use `GetTracking`'s `NextToken` to constrain the number of ad beacons returned to a player.

MediaTailor receives a `GetTracking` request. The response contains an ad with ID 9935407 and two beacons with `StartTimeInSeconds` values 52.286 and 48.332 seconds.

MediaTailor sends the JSON response with `NextToken` as follows:

```
{
"NextToken": JF57ITe48t1441mv7TmLKuZLroxDzfIslp6BiSNL1IJmzPVMDN0lqrBYycgMbKEb
"avails": [
  {
    "ads": [
      {
        "adId": "9935407",
        "adVerifications": [],
        "companionAds": [],
```

```
          "creativeId": "",
          "creativeSequence": "",
          "duration": "PT15S",
          "durationInSeconds": 15,
          "extensions": [],
          "mediaFiles": {
            "mediaFilesList": [],
            "mezzanine": ""
          },
          "startTime": "PT30S",
          "StartTimeInSeconds": 45,
          "trackingEvents": [
            {
              "beaconUrls": [
                "http://adserver.com/tracking?event=Impression "
              ],
              "duration": "PT0S",
              "durationInSeconds": 0,
              "eventId": "9935414",
              "eventType": "secondQuartile",
              "startTime": "PT52.286S",
              "StartTimeInSeconds": 52.286
            },
            {
              "beaconUrls": [
                "http://adserver.com/tracking?event=firstQuartile"
              ],
              "duration": "PT0S",
              "durationInSeconds": 0,
              "eventId": "9935412",
              "eventType": "firstQuartile",
              "startTime": "PT48.332S",
              "StartTimeInSeconds": 48.332
            }
          ],
          "vastAdId": ""
        }
      ],
      "startTime": "PT46.47S",
      "StartTimeInSeconds": 46.47
    }
  ]
}
```

On the next `GetTracking` request, MediaTailor responds with the `NextToken` value,: JF57ITe48t1441mv7TmLKuZLroxDzfIslp6BiSNL1IJmzPVMDN0lqrBYycgMbKEb.

MediaTailor responds with ads and beacons that match the `StartTimeInSeconds` that are set in the `NextToken` of the previous call.

Assume that now the response includes another ad with ID 9235407 in addition to the previous ad with ID 9935407. The beacons of ad ID 9235407 have `StartTimeInSecondss` 132.41 and 70.339.

MediaTailor iterates over all the beacons in the session to select the ones with `StartTimeInSeconds` greater than 52.286 seconds, which are beacon 3 and beacon 4 from the ad with ID 9235407:

```
{
  "NextToken": ZkfknvbfsdgfbsDFRdffg12EdffecFRvhjyjfhdfhnjtsg5SDGN
  "avails": [
    {
      "ads": [
        {
          "adId": "9235407",
          "adVerifications": [],
          "companionAds": [],
          "creativeId": "",
          "creativeSequence": "",
          "duration": "PT15.816S",
          "durationInSeconds": 19.716,
          "extensions": [],
          "mediaFiles": {
            "mediaFilesList": [],
            "mezzanine": ""
          },
          "startTime": "PT2M0S",
          "StartTimeInSeconds": 120.0,
          "trackingEvents": [
            {
              "beaconUrls": [
                "http://adserver.com/tracking?event=complete"
              ],
              "duration": "PT0S",
```

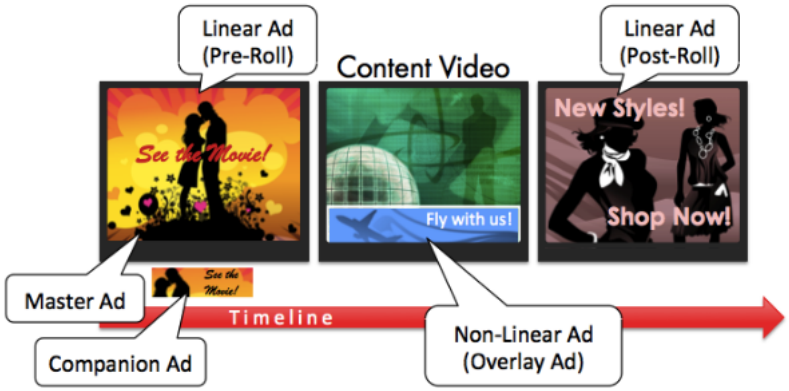
```
              "durationInSeconds": 0,
              "eventId": "8935414",
              "eventType": "firstQuartile",
              "startTime": "PT1M10.330S",
              "StartTimeInSeconds": 70.339
            },
            {
              "beaconUrls": [
                "http://adserver.com/tracking?event=thirdQuartile"
              ],
              "duration": "PT0S",
              "durationInSeconds": 0,
              "eventId": "8935412",
              "eventType": "secondQuartile",
              "startTime": "PT2M12.41S",
              "StartTimeInSeconds": 132.41
            }
          ],
          "vastAdId": ""
        },
      ],
      "startTime": "PT36.47S",
      "StartTimeInSeconds": 36.47
    }
  ]
}
```

Overlay ads

For live-streaming workflows where you want to increase monetization without interrupting the viewing experience with mid-roll ads, you can leverage your current AWS Elemental MediaTailor integration to guide an advertising format that is rendered client-side. This type of advertising is known as *overlay ads*. Overlay ads are non-linear video ads that appear in the form of 'L-band ads,' 'non-linear video ads,' 'picture-in-picture ads,' 'motion overlays', 'in-content advertising', or 'frame ads.'

MediaTailor detects a SCTE-35 marker with segmentation type `id=0x38` as an in-band signal for an overlay ad-insertion opportunity. The SCTE-35 marker causes MediaTailor to send a request to the Ad Decision Server (ADS), which then responds with a non-linear ad payload in the VAST response. MediaTailor parses the VAST response in order to support overlay ad insertion.

MediaTailor does not perform any stitching of linear ads, but rather signals to the player that there's a non-linear overlay ad available to play. This signaling allows the player to fetch and correlate the non-linear ads to play from the client-side tracking endpoint. The player then handles the display, reporting, and other tasks related to those ads. For example, the player's developer can use a device SDK from a vendor that supports overlay-ad formats. For more information about client-side tracking integrations, see Client-side ad-tracking integrations.



Topics

- Prerequisites for using overlay ads with MediaTailor
- Getting started using overlay ads with MediaTailor
- Logging and metrics for overlay ads in MediaTailor
- Billing for overlay ads in MediaTailor

Prerequisites for using overlay ads with MediaTailor

The following prerequisites apply when using overlay ads with MediaTailor:

- The workflow must be live, not video on demand (VOD).
- The Ad Decision Server (ADS) response must be configured to return only non-linear ads in the VAST response. MediaTailor ignores any linear ads for the purposes of ad stitching.
- The manifest must use a SCTE-35 time signal message with segmentation type `id=0x38` to invoke the overlay-ad feature.
- The streaming provider must have control of the client-device application and be integrated with the MediaTailor client-side tracking API.

Getting started using overlay ads with MediaTailor

This section explains how to get started using the overlay-ads feature of MediaTailor. You'll set up SCTE-35 signaling, configure Ad Decision Server (ADS) responses, and set up session-level control.

Topics

- Enabling overlay ads
- Tracking overlay ads with client-side metadata

Enabling overlay ads

MediaTailor support for overlay ads is enabled by default. A specific SCTE-35 ad-marker type in the manifest triggers the insertion of an overlay ad. Because some players might not support client-side rendering of overlay ads, you can disable the feature at the session level.

To disable overlay-ad support using HLS or DASH playback prefixes:

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

```
GET mediatailorURL/v1/master/hashed-account-id/origin-id/asset-id?
aws.overlayAvails=off
```

  - Example: DASH format

```
GET mediatailorURL/v1/master/hashed-account-id/origin-id/asset-id?
aws.overlayAvails=off
```

To disable overlay-ad support using the session-initialization prefix:

- On the player, construct a JSON message body for the session initialization request to MediaTailor:
  - To disable ad-overlay support, add an `overlays` object as a top-level key with a value of `off`. The default `overlays` value is on.

- (Optional) Provide any parameters that MediaTailor then passes to the ADS inside an `adParams` object. These parameters correspond to `[player_params.param]` settings in the ADS template URL of the MediaTailor configuration.

Example : HLS

```
POST master.m3u8
    {
       "adsParams": {
           "deviceType": "ipad"
       },
       "overlayAvails": "off"
    }
```

Example : DASH

```
POST manifest.mpd
    {
        "adsParams": {
           "deviceType": "androidmobile"
       },
       "overlayAvails": "off"
    }
```

Manifest signaling

MediaTailor trigger overlay-ads support when it sees a specific SCTE-35 marker in the manifest. The required signal is a splice command type 6, or time signal, that is a Provider Overlay Advertisement Start signal. This signal has a segmentation type id of `0x38`

The following example shows the `0x38` SCTE-35 marker in a JSON object.

```
{
  "tableId": 252,
  "selectionSyntaxIndicator": false,
  "privateIndicator": false,
  "sectionLength": 53,
  "protocolVersion": 0,
  "encryptedPacket": false,
  "encryptedAlgorithm": 0,
  "ptsAdjustment": 0,
```

```
  "cwIndex": 0,
  "tier": 4095,
  "spliceCommandLength": 5,
  "spliceCommandType": 6,
  "spliceCommand": {
    "specified": true,
    "pts": 1800392
  },
  "descriptorLoopLength": 31,
  "descriptors": [
    {
      "spliceDescriptorTag": 2,
      "descriptorLength": 29,
      "indentifier": "CUEI",
      "segmentationEventId": 158389361,
      "segmentationEventCancelIndicator": false,
      "programSegmentationFlag": true,
      "segmentationDurationFlag": true,
      "deliveryNotRestrictedFlag": false,
      "webDeliveryAllowedFlag": true,
      "noRegionalBlackoutFlag": true,
      "archiveAllowedFlag": true,
      "deviceResctrictions": 3,
      "segmentationDuration": 1350000,
      "segmentationUpidType": 9,
      "segmentationUpidLength": 7,
      "segmentationUpid": {
        "0": 111,
        "1": 118,
        "2": 101,
        "3": 114,
        "4": 108,
        "5": 97,
        "6": 121
      },
      "segmentationTypeId": 56,
      "segmentNum": 1,
      "segmentsExpected": 0
    }
  ],
  "crc": 2510422713
}
```

The following example shows the SCTE-35 signal represented as a binary (base 32/hexadecimal) value:

```
0xfc30350000000000000fff00506fe001b78c8001f021d435545490970d4717fdf00000dbba009076f7665726c617
```

The following examples shows the SCTE-35 marker in both HLS and DASH manifests.

Example : HLS manifest

```
#EXTM3U
#EXT-X-VERSION:6
#EXT-X-TARGETDURATION:7
#EXT-X-MEDIA-SEQUENCE:419
#EXT-X-DISCONTINUITY-SEQUENCE:3
#EXT-X-PROGRAM-DATE-TIME:2023-08-15T04:30:09.231Z
#EXTINF:6.02,
https://aws.cloudfront.net/media/asset1/index1_00007.ts
#EXT-X-DISCONTINUITY
#EXT-X-KEY:METHOD=NONE
#EXT-X-PROGRAM-DATE-TIME:2023-08-15T04:30:15.251Z
#EXTINF:6.0,
https://aws.cloudfront.net/media/asset1/index1_00001.ts
#EXT-X-PROGRAM-DATE-TIME:2023-08-15T04:30:21.251Z
#EXTINF:4.0,
https://aws.cloudfront.net/media/asset1/index1_00002.ts
#EXT-X-DISCONTINUITY
#EXT-X-DATERANGE:ID="1692073825251-30-1",START-
DATE="2023-08-15T04:30:25.251Z",DURATION=10.0,PLANNED-DURATION=10.0,SCTE35-
OUT=0xfc30350000000000000fff00506fe001b78c8001f021d435545490970d4717fdf00000dbba009076f766572
#EXT-X-PROGRAM-DATE-TIME:2023-08-15T04:30:25.251Z
#EXTINF:2.0,
https://aws.cloudfront.net/media/asset1/index1_00003.ts
#EXT-X-PROGRAM-DATE-TIME:2023-08-15T04:30:27.251Z
#EXTINF:6.0,
https://aws.cloudfront.net/media/asset1/index1_00004.ts
#EXT-X-PROGRAM-DATE-TIME:2023-08-15T04:30:33.251Z
#EXTINF:2.0,
https://aws.cloudfront.net/media/asset1/index1_00005.ts
#EXT-X-DISCONTINUITY
#EXT-X-PROGRAM-DATE-TIME:2023-08-15T04:30:35.251Z
#EXTINF:4.0,
https://aws.cloudfront.net/media/asset1/index1_00006.ts
#EXT-X-PROGRAM-DATE-TIME:2023-08-15T04:30:39.251Z
```

```
#EXTINF:6.02,
https://aws.cloudfront.net/media/asset1/index1_00007.ts
```

Example : DASH manifest

```
<?xml version="1.0"?>
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:scte35="urn:scte:scte35:2013:xml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 availabilityStartTime="2023-08-15T16:34:05.911Z" minBufferTime="PT30S"
 minimumUpdatePeriod="PT2S" profiles="urn:mpeg:dash:profile:isoff-live:2011"
 publishTime="2023-08-15T16:34:17.950Z" suggestedPresentationDelay="PT20S"
 timeShiftBufferDepth="PT1M30S" type="dynamic"
 xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/ittf/
PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd">
  <Period xmlns="urn:mpeg:dash:schema:mpd:2011" id="1692117245944_1" start="PT0.033S">
    <BaseURL>https://aws.cloudfront.net/out/v1/abc/123/def/</BaseURL>
    <EventStream schemeIdUri="urn:scte:scte35:2013:xml" timescale="90000">
      <Event duration="900000">
        <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="0" tier="4095">
          <scte35:TimeSignal>
            <scte35:SpliceTime ptsTime="0"/>
          </scte35:TimeSignal>
          <scte35:SegmentationDescriptor segmentNum="0" segmentationDuration="900000"
 segmentationEventCancelIndicator="false" segmentationEventId="1"
 segmentationTypeId="56" segmentsExpected="0" subSegmentNum="0"
 subSegmentsExpected="0">
            <scte35:SegmentationUpid segmentationUpidFormat="hexBinary"
 segmentationUpidType="14">63736f7665726c6179</scte35:SegmentationUpid>
          </scte35:SegmentationDescriptor>
        </scte35:SpliceInfoSection>
      </Event>
    </EventStream>
    <AdaptationSet bitstreamSwitching="true" mimeType="video/mp4"
 segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
      <Representation bandwidth="3000000" codecs="avc1.4D4028" frameRate="30/1"
 height="1080" id="1" width="1920">
        <SegmentTemplate initialization="../
cf684d31ec9e451ca98d2349989f6c0a/855c733eed20493ab3cc1100750bcf0b/
index_video_1_0_init.mp4" media="../
cf684d31ec9e451ca98d2349989f6c0a/855c733eed20493ab3cc1100750bcf0b/index_video_1_0_
$Number$.mp4" presentationTimeOffset="0" startNumber="1" timescale="30000">
          <SegmentTimeline>
```

```
            <S d="60000" r="6" t="1000"/>
            <S d="30000" t="421000"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
      <Representation bandwidth="2499968" codecs="avc1.4D4028" frameRate="30/1"
 height="1080" id="2" width="1920">
        <SegmentTemplate initialization="../
cf684d31ec9e451ca98d2349989f6c0a/855c733eed20493ab3cc1100750bcf0b/
index_video_2_0_init.mp4" media="../
cf684d31ec9e451ca98d2349989f6c0a/855c733eed20493ab3cc1100750bcf0b/index_video_2_0_
$Number$.mp4" presentationTimeOffset="0" startNumber="1" timescale="30000">
          <SegmentTimeline>
            <S d="60000" r="6" t="1000"/>
            <S d="30000" t="421000"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
      <Representation bandwidth="2200000" codecs="avc1.4D401F" frameRate="30/1"
 height="720" id="3" width="1280">
        <SegmentTemplate initialization="../
cf684d31ec9e451ca98d2349989f6c0a/855c733eed20493ab3cc1100750bcf0b/
index_video_3_0_init.mp4" media="../
cf684d31ec9e451ca98d2349989f6c0a/855c733eed20493ab3cc1100750bcf0b/index_video_3_0_
$Number$.mp4" presentationTimeOffset="0" startNumber="1" timescale="30000">
          <SegmentTimeline>
            <S d="60000" r="6" t="1000"/>
            <S d="30000" t="421000"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
    </AdaptationSet>
    <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
      <Label>Alternate Audio</Label>
      <Representation audioSamplingRate="48000" bandwidth="128000" codecs="mp4a.40.2"
 id="9">
        <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
        <SegmentTemplate initialization="../
cf684d31ec9e451ca98d2349989f6c0a/855c733eed20493ab3cc1100750bcf0b/
index_audio_9_0_init.mp4" media="../
cf684d31ec9e451ca98d2349989f6c0a/855c733eed20493ab3cc1100750bcf0b/index_audio_9_0_
$Number$.mp4" presentationTimeOffset="0" startNumber="1" timescale="48000">
          <SegmentTimeline>
```

```
            <S d="98304" t="0"/>
            <S d="96256" t="98304"/>
            <S d="95232" t="194560"/>
            <S d="96256" r="2" t="289792"/>
            <S d="95232" t="578560"/>
            <S d="46080" t="673792"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Ad Decision Server (ADS) response

The ADS response must contain one valid tracking event. At minimum, the tracking event can be an `Impression` tracking event. The tracking event should contain at least one `NonLinear` ad. This ad is the overlay ad, taking the form of a static, HTML, or iFrame resource.

```
<vmap AdBreak breaktype="linear" breakId="csoverlay"
```

If the VAST response is a VMAP with `breakType` of `nonlinear`, the avail metadata is inside the `nonLinearAvails` root object. If the VAST response is a VMAP with a `breakType` of `linear`, or is a plain VAST response without VMAP, the avail metadata is inside the `avails` root object.

The following VAST response is a wrapped VMAP response with a `breakType` value of `linear`.

In addition to the wrapped VMAP response, MediaTailor also supports a wrapped VMAP response with a `breakType` value of `nonlinear`, and a plain VAST response.

```
<?xml version="1.0" encoding="utf-8"?>
<vmap:VMAP xmlns:vmap="http://www.iab.net/vmap-1.0" version="1.0">
  <vmap:AdBreak breakType="linear" breakId="csoverlay">
    <vmap:AdSource allowMultipleAds="true" followRedirects="true" id="1">
      <vmap:VASTAdData>
        <VAST xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="3.0"
 xsi:noNamespaceSchemaLocation="vast.xsd">
          <Ad sequence="1">
            <InLine>
              <AdSystem>2.0</AdSystem>
              <AdTitle>2</AdTitle>
```

```
              <Impression><![CDATA[https://adserver.com/beacon=impression]]></
Impression>
              <Creatives>
                <Creative>
                  <NonLinearAds>
                    <NonLinear width="640" height="360" id="18">
                      <StaticResource creativeType="text/js_ref"><![CDATA[https://
client-side-ads.com/tags/static/ctv-generic/overlay001.json?iv_geo_country%3DUS%26]]></
StaticResource>
                    </NonLinear>
                  </NonLinearAds>
                </Creative>
              </Creatives>
            </InLine>
          </Ad>
        </VAST>
      </vmap:VASTAdData>
    </vmap:AdSource>
    <vmap:TrackingEvents>
      <vmap:Tracking event="breakStart"><![CDATA[https://adserver.com/
beacon=breakstartimpression]]></vmap:Tracking>
      <vmap:Tracking event="breakEnd"><![CDATA[https://adserver.com/
beacon=breakendimpression]]></vmap:Tracking>
    </vmap:TrackingEvents>
  </vmap:AdBreak>
</vmap:VMAP>
```

Example 1: DASH manifest source to MediaTailor

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:scte35="urn:scte:scte35:2013:xml"
 xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/
ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd" id="201"
 type="dynamic" publishTime="2022-11-07T19:59:05+00:00" minimumUpdatePeriod="PT2S"
 availabilityStartTime="2022-11-07T06:57:11.250000+00:00" minBufferTime="PT10S"
 suggestedPresentationDelay="PT20.000S" timeShiftBufferDepth="PT58.999S"
 profiles="urn:mpeg:dash:profile:isoff-live:2011">
  <Period start="PT46827.601S" id="0" duration="PT88.321S">
  ...
  </Period>
  <Period start="PT46915.922S" id="45" duration="PT6.006S">
    <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2014:xml+bin">
```

```
    <Event duration="540000" id="144">
        <scte35:Signal>
            <scte35:Binary>SCTE35-binary</scte35:Binary>
        </scte35:Signal>
    </Event>
    </EventStream>
    ...
  </Period>
  <Period start="PT46921.928S" id="49">
  ...
  </Period>
</MPD>
```

Example 2: MediaTailor personalized DASH manifest containing an ad ID decoration

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:scte35="urn:scte:scte35:2013:xml"
 xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/
ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd" id="201"
 type="dynamic" publishTime="2022-11-07T19:59:05+00:00" minimumUpdatePeriod="PT2S"
 availabilityStartTime="2022-11-07T06:57:11.250000+00:00" minBufferTime="PT10S"
 suggestedPresentationDelay="PT20.000S" timeShiftBufferDepth="PT58.999S"
 profiles="urn:mpeg:dash:profile:isoff-live:2011">
  <Period start="PT46827.601S" id="0" duration="PT88.321S">
  ...
  </Period>
  <Period start="PT46915.922S" id="45" duration="PT6.006S">
  <EventStream schemeIdUri="urn:sva:advertising-wg:ad-id-signaling" timescale="90000">
    <Event presentationTime="13500000" duration="1351350">
    <![CDATA[{"version": 1,"identifiers": [{"scheme":
 "urn:smpte:ul:060E2B34.01040101.01200900.00000000","value": "adId","ad_position":
 "adId", "ad_type":"overlay","creative_id": "creativeId","tracking_uri":
 "trackingUri"}]}]]></Event>
  </EventStream>
  ...
  </Period>
  <Period start="PT46921.928S" id="49">
  ...
  </Period>
</MPD>
```

Tracking overlay ads with client-side metadata

MediaTailor places the overlay ads in the `nonLinearAdsList` of the avail. The MediaTailor client-side tracking API has two root objects, called `avails` and `nonLinearAvails`. If the VAST response is a VMAP with `breakType` of `nonlinear`, the avail metadata is inside the `nonLinearAvails` root object. If the VAST response is a VMAP with a `breakType` of `linear`, or is a plain VAST response without VMAP, the avail metadata is inside the `avails` root object.

For more information about client-side tracking, see Client-side ad tracking.

The following example shows a plain VAST response or VMAP response with a `breakType` value of `linear`.

```
{
  "avails": [
    {
      "adBreakTrackingEvents": [
        {
          "beaconUrls": [
            "https://adserver.com/beacon=breakstartimpression"
          ],
          "eventType": "breakStart"
        },
        {
          "beaconUrls": [
            "https://adserver.com/beacon=breakendimpression"
          ],
          "eventType": "breakEnd"
        }
      ],
      "adMarkerDuration": null,
      "ads": [],
      "availId": "828",
      "availProgramDateTime": null,
      "duration": "PT0S",
      "durationInSeconds": 0,
      "meta": null,
      "nonLinearAdsList": [
        {
          "extensions": null,
          "nonLinearAdList": [
            {
              "adId": "",
```

```
              "adParameters": null,
              "adSystem": "2.0",
              "adTitle": "2",
              "apiFramework": null,
              "clickThrough": null,
              "clickTracking": null,
              "clickTrackingId": null,
              "creativeAdId": "",
              "creativeId": "18",
              "creativeSequence": "",
              "duration": null,
              "durationInSeconds": 0,
              "expandedHeight": null,
              "expandedWidth": null,
              "height": "360",
              "htmlResource": null,
              "iFrameResource": null,
              "maintainAspectRatio": false,
              "minSuggestedDuration": null,
              "scalable": false,
              "staticResource": "https://client-side-ads.com/tags/static/ctv-generic/
overlay001.json?iv_geo_country%3DUS%26",
              "staticResourceCreativeType": "text/js_ref",
              "width": "640"
            }
          ],
          "trackingEvents": [
            {
              "beaconUrls": [
                "https://adserver.com/beacon=impression"
              ],
              "duration": null,
              "durationInSeconds": 0,
              "eventId": null,
              "eventProgramDateTime": null,
              "eventType": "impression",
              "startTime": null,
              "startTimeInSeconds": 0
            }
          ]
        }
      ],
      "startTime": "PT1M46.08S",
      "startTimeInSeconds": 106.08
```

```
    }
  ],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": null,
  "nonLinearAvails": []
}
```

The following example shows a plain VMAP response with a `breakType` value of `nonlinear`.

```
{
  "avails": [],
  "dashAvailabilityStartTime": null,
  "hlsAnchorMediaSequenceNumber": null,
  "nextToken": null,
  "nonLinearAvails": [
    {
      "adBreakTrackingEvents": [
        {
          "beaconUrls": [
            "https://adserver.com/beacon=breakstartimpression"
          ],
          "eventType": "breakStart"
        },
        {
          "beaconUrls": [
            "https://adserver.com/beacon=breakendimpression"
          ],
          "eventType": "breakEnd"
        }
      ],
      "adMarkerDuration": null,
      "ads": [],
      "availId": "828",
      "availProgramDateTime": null,
      "duration": "PT0S",
      "durationInSeconds": 0,
      "meta": null,
      "nonLinearAdsList": [
        {
          "extensions": null,
          "nonLinearAdList": [
            {
```

```
              "adId": "",
              "adParameters": null,
              "adSystem": "2.0",
              "adTitle": "2",
              "apiFramework": null,
              "clickThrough": null,
              "clickTracking": null,
              "clickTrackingId": null,
              "creativeAdId": "",
              "creativeId": "18",
              "creativeSequence": "",
              "duration": null,
              "durationInSeconds": 0,
              "expandedHeight": null,
              "expandedWidth": null,
              "height": "360",
              "htmlResource": null,
              "iFrameResource": null,
              "maintainAspectRatio": false,
              "minSuggestedDuration": null,
              "scalable": false,
              "staticResource": "https://client-side-ads.com/tags/static/ctv-generic/
overlay001.json?iv_geo_country%3DUS%26",
              "staticResourceCreativeType": "text/js_ref",
              "width": "640"
            }
          ],
          "trackingEvents": [
            {
              "beaconUrls": [
                "https://adserver.com/beacon=impression"
              ],
              "duration": null,
              "durationInSeconds": 0,
              "eventId": null,
              "eventProgramDateTime": null,
              "eventType": "impression",
              "startTime": null,
              "startTimeInSeconds": 0
            }
          ]
        }
      ],
      "startTime": "PT1M46.08S",
```

```
      "startTimeInSeconds": 106.08
    }
  ]
}
```

Logging and metrics for overlay ads in MediaTailor

This section explains logging and metrics for overlay ads in MediaTailor. For more information about setting up logging, see Monitoring and tagging AWS Elemental MediaTailor resources.

Topics

- CloudWatch logs
- CloudWatch metrics

CloudWatch logs

CloudWatch collects the following log information about overlay ads:

- `VAST_RESPONSE` - Shows information about the non-linear ads list.
- `FILLED_PROVIDER_OVERLAY` - Shows information about the non-linear ads.

> **Note**
>
> The `RAW_ADS_RESPONSE` is an optional event that shows the original response from the ADS. Using this event is especially helpful in a staging and testing environment. To enable this event on a configuration or account, submit a ticket to AWS Support.

CloudWatch metrics

MediaTailor collects overlay ad metrics separately from other ADS metrics. MediaTailor collects these metrics after successfully fetching the ads from the ADS. You don't have to poll the `GetTracking` API to collect the metrics.

The following table describes CloudWatch metrics for overlay ads:

Metric	Description
AdDecisionServer.OverlayAds	The count of overlay ads included in the ADS responses within the CloudWatch time period that you specified.
AdDecisionServer.OverlayErrors	The number of non-HTTP 200 status code responses, empty responses, and timed-out responses that MediaTailor received from the ADS within the CloudWatch time period that you specified.
AdDecisionServer.OverlayFilled	The number of avails that were successfully filled with at least one overlay ad: • 1 - There's at least one valid ad. • 0 - Either MediaTailor didn't get any overlay ads, or there was some other failure. SampleCount tracks the number of avails filled. Sum tracks the number of successfully filled overlay avails.
AdDecisionServer.OverlayMinSuggestedDuration	The sum of minSuggestedDuration durations, in milliseconds, of all ads that MediaTailor received from the ADS within the CloudWatch time period that you specified. If minSuggestedDuration isn't specified, the duration shown is the planned duration.
AdDecisionServer.OverlayLatency	The response time, in milliseconds, for requests that MediaTailor makes to the ADS.

Metric	Description
`AdDecisionServer.OverlayTimeouts`	The number of timed-out requests to the ADS in the CloudWatch time period that you specified.
`AdsBilled`	For more information about ads billed, see Billing for overlay ads in MediaTailor.
`Avail.*`	Because MediaTailor doesn't do any planning for overlay ads, CloudWatch doesn't show any `Avail.X` metrics.
`SkippedReason.*`	Because MediaTailor doesn't do any planning for overlay ads, CloudWatch doesn't show any `SkippedReason.X` metrics.

Billing for overlay ads in MediaTailor

MediaTailor bills customers based on the number of non-linear ads in the ADS response. This number includes non-linear ads that extend past the break duration. After MediaTailor fills the avail, it bills for the ads it filled.

For prefetch workflows, MediaTailor does not bill for ads when retrieving the prefetch, but rather, when it sees a compatible ad avail in the consumption window for that session.

For additional billing information, see https://aws.amazon.com/mediatailor/pricing/.

Ad ID decoration

AWS Elemental MediaTailor performs server side ad stitching when transitioning from content to ad breaks. MediaTailor can condition the manifest with metadata associated with the ads that have been stitched. Doing so can provide the following benefits:

- *Video start time* (VST) improves
- MediaTailor can support a hybrid model of server side ad insertion and server guided ad insertion
- Server side sessions can build playback timelines with ad position markers

- For client side sessions that already build playback timelines with the MediaTailor API, session VST improves, as the session does not rely on calling the tracking API to build the timeline
- It's possible to leverage MediaTailor for server side ad insertion as well as client side rendered ads displayed in-scene. This way, a player's software development kit (SDK) doesn't need to have a separate integration to call ad serving entities directly for client-side ads. MediaTailor can vend the ads through the manifest and the client-side tracking API.

There are standards for associating each creative ad asset with a unique identifier. This association allows advertisers, agencies, vendors, and publishers to relate a creative ad asset across their independent workflows. As metrics and monitoring of streams continue to improve and more distributors utilize server-based insertion architectures, the need arises to accurately communicate the identifiers assigned to individual creative assets within an interleaved/stitched presentation, such as within the personalized manifest.

Topics

- Enabling ad ID signaling for sessions
- Manifests and ad metadata insertion
- Ad Decision Server (ADS) interactions
- Client-side tracking API

Enabling ad ID signaling for sessions

The ad ID signaling feature must be enabled during session initialization. The process to enable the feature differs from creating sessions using the HLS/DASH playback prefix (implicit session initialization), versus the session initialization prefix (explicit session initialization).

To enable ad ID for the session using HLS/DASH playback prefixes

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

```
GET <mediatailorURL>/v1/master/<hashed-account-id>/<origin-id>/<asset-id>?
aws.adSignalingEnabled=true
```

  - Example: DASH format

```
GET <mediatailorURL>/v1/dash/<hashed-account-id>/<origin-id>/<asset-id>?
aws.adSignalingEnabled=true
```

To enable ad ID for the session using the session initialization prefix

- On the player, construct a JSON message body for the session initialization request to MediaTailor:
  - Inside an `adsParams` object, provide any parameters that MediaTailor should pass to the ADS. These parameters correspond to `[player_params.param]` settings in the ADS template URL of the MediaTailor configuration.
  - To enable ad ID signaling, add an `adSignaling` object as a top level object, and inside, add a parameter called `enabled` and value of `true`. The default `adSignaling` value is `disabled`.
  - Example: HLS format

```
POST master.m3u8
    {
       "adsParams": {
           "deviceType": "ipad"
       },
       "adSignaling": {
           "enabled": "true"
       }
    }
```

  - Example: DASH format

```
POST manifest.mpd
    {
        "adsParams": {
           "deviceType": "ipad"
       },
       "adSignaling": {
            "enabled": "true"
        }
    }
```

Manifests and ad metadata insertion

During the ad stitching process, MediaTailor adds to the manifest the unique ID associated with each creative being stitched. MediaTailor obtains the unique ID of the creative from the `id` attribute value of that creative in the VAST response. If the creative lacks an ID attribute value, MediaTailor will publish an empty value (`id=""`).

MediaTailor uses an in-manifest metadata signal to decouple dependencies between the client tracking API for ad creative metadata and timing/positioning within the overall timeline. This decoupling reduces playback latency (especially in VOD scenarios), where the player's user interface (UI) renders ad break positions in the timeline prior to initializing playback.

The added metadata takes the following forms:

- For HLS manifests, the added metadata takes the form of `DATERANGE` tags for each ad in the avail period.
- For DASH manifests, the added metadata takes the form of an `Event` element within each ad period.

The following JSON message body shows an example VAST response:

```
{
  "version": 1,
  "identifiers": [
    {
      "scheme": "urn:smpte:ul:060E2B34.01040101.01200900.00000000",
      "value": "creativeId",
      "ad_position": "adId",
      "ad_type": "adType",
      "tracking_uri": "trackingUri",
      "custom_vast_data":"customVastData"
    }
  ]
}
```

In the preceding example:

- *`creativeId`* is the `Id` attribute value of the `Creative` element for the ad
- *`adId`* is either the HLS sequence number associated with the beginning of the ad, or the DASH period ID of the ad

- *adType* is either `avail` or `overlay`, based on the VAST response
- *trackingUri* is the relative tracking endpoint for the MediaTailor session, in the format `../../../../tracking/`*hashed-account-id*`/`*origin-id*`/`*session-id*
- *customVastData* is a value that MediaTailor extracts from the `creative_signaling` VAST extension. MediaTailor uses the contents of the CDATA node, if present. See the Ad Decision Server (ADS) interactions section for more details and a sample VAST response.

Personalizing HLS manifests with ad metadata

For a live HLS stream, MediaTailor only adds metadata when the stream contains `PROGRAM-DATE-TIME` tags, at least once per manifest duration. For a video on demand (VOD) stream, MediaTailor adds `PROGRAM-DATE-TIME` to at least one segment in the personalized manifest, where the start time for each VOD asset is epoch zero (`1970-01-01T00:00:00Z`). If the origin manifest has existing `PROGRAM-DATE-TIME` content, then MediaTailor preserves that content.

MediaTailor personalizes the manifest with creatives returned by the Ad Decision Server (ADS). For each ad, MediaTailor also includes a `DATERANGE` tag that spans the duration of the ad. The `DATERANGE` tag format is similar to that described in the section Ad creative signaling in DASH and HLS in the 2023 version of the *SVA technical publication*.

The `DATERANGE` that MediaTailor generates has unique ID values. To ensure uniqueness (given the guidelines specified in Mapping SCTE-35 into EXT-X-DATERANGE), MediaTailor couples the `MEDIA-SEQUENCE` number of the *first* ad segment of the avail with the sequence number of the ad within the avail.

For underfilled ad breaks on configurations that have slate enabled, MediaTailor appends the slate segments to the end of the avail, separated by a `DISCONTINUITY` tag, but without any `DATERANGE` metadata.

For each ad stitched into the personalized manifest, MediaTailor adds the creative metadata, represented as base64-encoded data in a custom `DATERANGE` tag.

Example Linear HLS origin (`#EXT-X-CUE-OUT`):

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:7
#EXT-X-MEDIA-SEQUENCE:398
#EXT-X-PROGRAM-DATE-TIME:2023-02-10T19:20:01.397Z
```

```
#EXTINF:6.006,
index_1_398.ts?m=1676054627
#EXTINF:5.873,
index_1_399.ts?m=1676054627
#EXT-OATCLS-SCTE35:/DAlAAAAAyiYAP/wFAUAAAACf+//jPl97P4AUmNiAAEBAQAAse4/gA==
#EXT-X-CUE-OUT:59.993
#EXTINF:6.139,
index_1_400.ts?m=1676054627
#EXT-X-CUE-OUT-CONT:ElapsedTime=6.139,Duration=59.993,SCTE35=/DAlAAAAAyiYAP/wFAUAAAACf
+//jPl97P4AUmNiAAEBAQAAse4/gA==
#EXTINF:6.006,
index_1_401.ts?m=1676054627
#EXT-X-CUE-OUT-CONT:ElapsedTime=12.145,Duration=59.993,SCTE35=/DAlAAAAAyiYAP/wFAUAAAACf
+//jPl97P4AUmNiAAEBAQAAse4/gA==
#EXTINF:6.006,
index_1_402.ts?m=1676054627
#EXT-X-CUE-OUT-CONT:ElapsedTime=18.151,Duration=59.993,SCTE35=/DAlAAAAAyiYAP/wFAUAAAACf
+//jPl97P4AUmNiAAEBAQAAse4/gA==
#EXTINF:6.006,
index_1_403.ts?m=1676054627
#EXT-X-CUE-OUT-CONT:ElapsedTime=24.157,Duration=59.993,SCTE35=/DAlAAAAAyiYAP/wFAUAAAACf
+//jPl97P4AUmNiAAEBAQAAse4/gA==
#EXTINF:6.006,
index_1_404.ts?m=1676054627
#EXT-X-CUE-OUT-CONT:ElapsedTime=30.163,Duration=59.993,SCTE35=/DAlAAAAAyiYAP/wFAUAAAACf
+//jPl97P4AUmNiAAEBAQAAse4/gA==
#EXTINF:6.006,
index_1_405.ts?m=1676054627
#EXT-X-CUE-OUT-CONT:ElapsedTime=36.169,Duration=59.993,SCTE35=/DAlAAAAAyiYAP/wFAUAAAACf
+//jPl97P4AUmNiAAEBAQAAse4/gA==
#EXTINF:6.006,
index_1_406.ts?m=1676054627
#EXT-X-CUE-OUT-CONT:ElapsedTime=42.175,Duration=59.993,SCTE35=/DAlAAAAAyiYAP/wFAUAAAACf
+//jPl97P4AUmNiAAEBAQAAse4/gA==
#EXTINF:6.006,
index_1_407.ts?m=1676054627
#EXT-X-CUE-OUT-CONT:ElapsedTime=48.181,Duration=59.993,SCTE35=/DAlAAAAAyiYAP/wFAUAAAACf
+//jPl97P4AUmNiAAEBAQAAse4/gA==
#EXTINF:6.006,
index_1_408.ts?m=1676054627
#EXT-X-CUE-OUT-CONT:ElapsedTime=54.187,Duration=59.993,SCTE35=/DAlAAAAAyiYAP/wFAUAAAACf
+//jPl97P4AUmNiAAEBAQAAse4/gA==
#EXTINF:5.806,
index_1_409.ts?m=1676054627
```

```
#EXT-X-CUE-IN
#EXTINF:6.206,
index_1_410.ts?m=1676054627
#EXTINF:6.006,
index_1_411.ts?m=1676054627
#EXTINF:6.006,
index_1_412.ts?m=1676054627
```

Example Linear HLS origin (#EXT-X-DATERANGE):

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:7
#EXT-X-MEDIA-SEQUENCE:25
#EXT-X-PROGRAM-DATE-TIME:2023-02-10T19:19:53.389Z
#EXTINF:6.006,
index_1_25.ts?m=1676056675
#EXTINF:6.006,
index_1_26.ts?m=1676056675
#EXTINF:6.006,
index_1_27.ts?m=1676056675
#EXTINF:1.869,
index_1_28.ts?m=1676056675
#EXT-X-DATERANGE:ID="2",START-DATE="2023-02-10T19:20:13.276Z",PLANNED-
DURATION=59.993,SCTE35-
OUT=0xFC3025000000003289800FFF01405000000027FEFFF8CF97DECFE00526362000101010000B1EE3F80
#EXTINF:6.139,
index_1_29.ts?m=1676056675
#EXTINF:6.006,
index_1_30.ts?m=1676056675
#EXTINF:6.006,
index_1_31.ts?m=1676056675
#EXTINF:6.006,
index_1_32.ts?m=1676056675
#EXTINF:6.006,
index_1_33.ts?m=1676056675
#EXTINF:6.006,
index_1_34.ts?m=1676056675
#EXTINF:6.006,
index_1_35.ts?m=1676056675
#EXTINF:6.006,
index_1_36.ts?m=1676056675
#EXTINF:6.006,
```

```
index_1_37.ts?m=1676056675
#EXTINF:5.806,
index_1_38.ts?m=1676056675
#EXT-X-DATERANGE:ID="2",START-DATE="2023-02-10T19:20:13.276Z",END-
DATE="2023-02-10T19:21:13.269Z",DURATION=59.993
#EXTINF:6.206,
index_1_39.ts?m=1676056675
#EXTINF:6.006,
index_1_40.ts?m=1676056675
```

Example Linear HLS personalized manifest (with creative ad signaling):

The DATERANGE that MediaTailor generates has unique ID values. To ensure uniqueness (given the guidelines specified in Mapping SCTE-35 into EXT-X-DATERANGE), MediaTailor couples the MEDIA-SEQUENCE number of the *first* ad segment of the avail with the sequence number of the ad within the avail.

In the following example, MediaTailor concatenates MEDIA-SEQUENCE 421 with the ad position number.

```
#EXTM3U
#EXT-X-VERSION:6
#EXT-X-TARGETDURATION:7
#EXT-X-MEDIA-SEQUENCE:418
#EXT-X-DISCONTINUITY-SEQUENCE:5
#EXT-X-PROGRAM-DATE-TIME:2023-02-10T19:19:55.391Z
#EXTINF:6.006,
https://d3fch9e2fcarly.cloudfront.net/out/v1/1cc7058242a74fdd8aea14e22a9b4131/
index_1_397.ts?m=1676054627
#EXTINF:6.006,
https://d3fch9e2fcarly.cloudfront.net/out/v1/1cc7058242a74fdd8aea14e22a9b4131/
index_1_398.ts?m=1676054627
#EXTINF:5.873,
https://d3fch9e2fcarly.cloudfront.net/out/v1/1cc7058242a74fdd8aea14e22a9b4131/
index_1_399.ts?m=1676054627
#EXT-X-DISCONTINUITY
#EXT-X-PROGRAM-DATE-TIME:2023-02-10T19:19:55.391Z
#EXT-X-DATERANGE:ID="421-1",CLASS="urn:sva:advertising-wg:ad-id-signaling",START-
DATE=2019-01-01T00:02:30.000Z,DURATION=15.015,X-AD-CREATIVE-SIGNALING="base64JSON"
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056813
#EXTINF:2.002,
```

```
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056814
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056815
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056816
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056817
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056818
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056819
#EXTINF:1.001,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056820
#EXT-X-DISCONTINUITY
#EXT-X-PROGRAM-DATE-TIME:2023-02-10T19:19:55.391Z
#EXT-X-DATERANGE:ID="421-1",START-DATE="2023-02-10T19:36:13.435Z",END-
DATE="2023-02-10T19:36:43.432Z",DURATION=15.015
#EXT-X-DATERANGE:ID="421-2",CLASS="urn:sva:advertising-wg:ad-id-signaling",START-
DATE=2019-01-01T00:02:30.000Z,DURATION=15.015,X-AD-CREATIVE-SIGNALING="base64JSON"
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056821
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056822
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056823
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056824
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056825
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056826
```

```
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056827
#EXTINF:1.001,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056828
#EXT-X-DISCONTINUITY
#EXT-X-PROGRAM-DATE-TIME:2023-02-10T19:19:55.391Z
#EXT-X-DATERANGE:ID="421-2",START-DATE="2023-02-10T19:36:13.435Z",END-
DATE="2023-02-10T19:36:43.432Z",DURATION=15.015
#EXT-X-DATERANGE:ID="421-3",CLASS="urn:sva:advertising-wg:ad-id-signaling",START-
DATE=2019-01-01T00:02:30.000Z,DURATION=15.015,X-AD-CREATIVE-SIGNALING="base64JSON"
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056829
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056830
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056831
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056832
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056833
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056834
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056835
#EXTINF:1.001,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056836
#EXT-X-DISCONTINUITY
#EXT-X-PROGRAM-DATE-TIME:2023-02-10T19:19:55.391Z
#EXT-X-DATERANGE:ID="421-3",START-DATE="2023-02-10T19:36:13.435Z",END-
DATE="2023-02-10T19:36:43.432Z",DURATION=29.997
#EXT-X-DATERANGE:ID="421-4",CLASS="urn:sva:advertising-wg:ad-id-signaling",START-
DATE=2019-01-01T00:02:30.000Z,DURATION=15.015,X-AD-CREATIVE-SIGNALING="base64JSON"
#EXTINF:2.002,
```

```
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056837
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056838
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056839
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056840
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056841
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056842
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056843
#EXTINF:1.001,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/9e178fa9-
dce5-4248-83d2-5b5d98b019bf/0/1676056844
#EXT-X-DISCONTINUITY
#EXT-X-DATERANGE:ID="421-4",START-DATE="2023-02-10T19:36:13.435Z",END-
DATE="2023-02-10T19:36:43.432Z",DURATION=15.015
#EXTINF:6.206,
https://d3fch9e2fcarly.cloudfront.net/out/v1/1cc7058242a74fdd8aea14e22a9b4131/
index_1_410.ts?m=1676054627
#EXTINF:6.006,
https://d3fch9e2fcarly.cloudfront.net/out/v1/1cc7058242a74fdd8aea14e22a9b4131/
index_1_411.ts?m=1676054627
```

Example VOD HLS origin (with SCTE signals):

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:7
#EXT-X-MEDIA-SEQUENCE:1
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:6,
index_720p1500k_00001.ts
#EXTINF:6,
```

```
index_720p1500k_00002.ts
#EXTINF:6,
index_720p1500k_00003.ts
#EXTINF:6,
index_720p1500k_00004.ts
#EXTINF:6,
index_720p1500k_00005.ts
#EXT-X-CUE-OUT:0
#EXT-X-CUE-IN
#EXTINF:6,
index_720p1500k_00006.ts
#EXTINF:6,
index_720p1500k_00007.ts
#EXTINF:6,
index_720p1500k_00008.ts
#EXTINF:6,
index_720p1500k_00009.ts
#EXTINF:6,
index_720p1500k_00010.ts
#EXTINF:6,
index_720p1500k_00011.ts
#EXTINF:6,
index_720p1500k_00012.ts
```

Example VOD HLS origin:

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:7
#EXT-X-MEDIA-SEQUENCE:1
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:6,
index_720p1500k_00001.ts
#EXTINF:6,
index_720p1500k_00002.ts
#EXTINF:6,
index_720p1500k_00003.ts
#EXTINF:6,
index_720p1500k_00004.ts
#EXTINF:4,
index_720p1500k_00005.ts
#EXTINF:2,
index_720p1500k_00006.ts
```

```
#EXTINF:6,
index_720p1500k_00007.ts
#EXTINF:6,
index_720p1500k_00008.ts
#EXTINF:6,
index_720p1500k_00009.ts
#EXTINF:6,
index_720p1500k_00010.ts
#EXTINF:6,
index_720p1500k_00011.ts
#EXTINF:6,
index_720p1500k_00012.ts
```

Example VOD HLS personalized manifest:

MediaTailor adds PROGRAM-DATE-TIME to VOD manifests in order to use them as anchors for the HLS DATERANGE elements that indicate ad positions.

The DATERANGE that MediaTailor generates has unique ID values. To ensure uniqueness (given the guidelines specified in Mapping SCTE-35 into EXT-X-DATERANGE), MediaTailor couples the MEDIA-SEQUENCE number of the *first* ad segment of the avail with the sequence number of the ad within the avail.

In the following example, MediaTailor concatenates MEDIA-SEQUENCE 421 with the ad position number.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-TARGETDURATION:7
#EXT-X-MEDIA-SEQUENCE:1
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-PROGRAM-DATE-TIME:1970-01-01T00:00:00Z
#EXTINF:6.0,
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00001.ts
#EXTINF:6.0,
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00002.ts
#EXTINF:6.0,
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00003.ts
#EXTINF:6.0,
```

```
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00004.ts
#EXTINF:4.0,
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00005.ts
#EXT-X-DISCONTINUITY
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/28
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/29
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/30
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/31
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/32
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/33
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/34
#EXTINF:1.001,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/35
#EXT-X-DISCONTINUITY
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/36
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/37
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/38
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/39
#EXTINF:2.002,
```

```
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/40
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/41
#EXTINF:2.002,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/42
#EXTINF:1.001,
../../../../segment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/vod-
variations/9810d863-8736-45fa-866e-be6d2c2bfa20/0/43
#EXT-X-DISCONTINUITY
#EXTINF:2.0,
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00006.ts
#EXTINF:6.0,
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00007.ts
#EXTINF:6.0,
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00008.ts
#EXTINF:6.0,
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00009.ts
#EXTINF:6.0,
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00010.ts
#EXTINF:6.0,
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00011.ts
#EXTINF:6.0,
https://d3fch9e2fcarly.cloudfront.net/cunsco-media/SKO-22/asset-1/hls/
index_720p1500k_00012.ts
#EXT-X-ENDLIST
#EXT-X-DATERANGE:ID="5-1",START-DATE="1970-01-01T00:00:28.000Z",END-
DATE="1970-01-01T00:00:43.015Z",DURATION=15.015
#EXT-X-DATERANGE:ID="5-2",START-DATE="1970-01-01T00:00:43.015Z",END-
DATE="1970-01-01T00:00:58.030Z",DURATION=15.01
```

Personalizing DASH manifests with ad metadata

MediaTailor personalizes the manifest with creatives returned by the Ad Decision Server (ADS). For each ad, MediaTailor also includes an `EventStream` element that spans the duration of the ad.

The Event element format is similar to that described in the section Ad creative signaling in DASH and HLS in the 2023 version of the *SVA technical publication*.

For underfilled ad breaks on configurations that have slate enabled, MediaTailor appends the slate period to the end of the avail period, but without any EventStream metadata

For each ad stitched into the personalized manifest, MediaTailor adds the creative metadata, represented as a CDATA element within an Event element.

Example Linear DASH origin (Inline SCTE attributes):

```
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:scte35="urn:scte:scte35:2013:xml"
 xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/
ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd" id="201"
 type="dynamic" publishTime="2023-02-10T21:08:40+00:00" minimumUpdatePeriod="PT6S"
 availabilityStartTime="2023-02-09T22:47:05.865000+00:00" minBufferTime="PT10S"
 suggestedPresentationDelay="PT20.000S" timeShiftBufferDepth="PT88.999S"
 profiles="urn:mpeg:dash:profile:isoff-live:2011">
  <Period start="PT80141.456S" id="104" duration="PT304.103S">
    <AdaptationSet id="1485523442" mimeType="video/mp4" segmentAlignment="true"
 startWithSAP="1" subsegmentAlignment="true" subsegmentStartsWithSAP="1"
 bitstreamSwitching="true">
      <SegmentTemplate timescale="60000" media="index_video_$RepresentationID$_0_
$Number$.mp4?m=1676062374" initialization="index_video_$RepresentationID$_0_init.mp4?
m=1676062374" startNumber="151" presentationTimeOffset="4808487386">
        <SegmentTimeline>
          <S t="4824975858" d="360360" r="3"/>
          <S t="4826417298" d="316316"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="1" width="960" height="540" frameRate="30000/1001"
 bandwidth="1800000" codecs="avc1.4D401F"/>
      <Representation id="3" width="640" height="360" frameRate="30000/1001"
 bandwidth="1200000" codecs="avc1.4D401E"/>
      <Representation id="5" width="480" height="270" frameRate="30000/1001"
 bandwidth="800000" codecs="avc1.4D4015"/>
    </AdaptationSet>
    <AdaptationSet id="1377232898" mimeType="audio/mp4" segmentAlignment="0"
 lang="eng">
      <Label>eng</Label>
```

```
      <SegmentTemplate timescale="48000" media="index_audio_$RepresentationID$_0_
$Number$.mp4?m=1676062374" initialization="index_audio_$RepresentationID$_0_init.mp4?
m=1676062374" startNumber="151" presentationTimeOffset="3846790126">
        <SegmentTimeline>
          <S t="3859981294" d="287744"/>
          <S t="3860269038" d="288768"/>
          <S t="3860557806" d="287744"/>
          <S t="3860845550" d="288768"/>
          <S t="3861134318" d="252928"/>
        </SegmentTimeline>
        </SegmentTemplate>
        <Representation id="2" bandwidth="193007" audioSamplingRate="48000"
 codecs="mp4a.40.2">
        <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
      <Representation id="4" bandwidth="193007" audioSamplingRate="48000"
 codecs="mp4a.40.2">
        <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
      <Representation id="6" bandwidth="193007" audioSamplingRate="48000"
 codecs="mp4a.40.2">
        <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
    </AdaptationSet>
    <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
 value="2023-02-10T21:02:31.007Z"/>
  </Period>
  <Period start="PT80445.560S" id="155" duration="PT44.978S">
    <EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
      <Event duration="4048044">
        <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="207000"
 tier="4095">
          <scte35:SpliceInsert spliceEventId="111" spliceEventCancelIndicator="false"
 outOfNetworkIndicator="true" spliceImmediateFlag="false" uniqueProgramId="1"
 availNum="1" availsExpected="1">
            <scte35:Program>
              <scte35:SpliceTime ptsTime="7239893422"/>
            </scte35:Program>
            <scte35:BreakDuration autoReturn="true" duration="4048044"/>
          </scte35:SpliceInsert>
        </scte35:SpliceInfoSection>
```

```
      </Event>
    </EventStream>
    <AdaptationSet id="1485523442" mimeType="video/mp4" segmentAlignment="true"
 startWithSAP="1" subsegmentAlignment="true" subsegmentStartsWithSAP="1"
 bitstreamSwitching="true">
      <SegmentTemplate timescale="60000" media="index_video_$RepresentationID$_0_
$Number$.mp4?m=1676062374" initialization="index_video_$RepresentationID$_0_init.mp4?
m=1676062374" startNumber="156" presentationTimeOffset="4826733614">
        <SegmentTimeline>
          <S t="4826733614" d="284284"/>
          <S t="4827017898" d="360360" r="5"/>
          <S t="4829180058" d="252252"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="1" width="960" height="540" frameRate="30000/1001"
 bandwidth="1800000" codecs="avc1.4D401F"/>
      <Representation id="3" width="640" height="360" frameRate="30000/1001"
 bandwidth="1200000" codecs="avc1.4D401E"/>
      <Representation id="5" width="480" height="270" frameRate="30000/1001"
 bandwidth="800000" codecs="avc1.4D4015"/>
    </AdaptationSet>
    <AdaptationSet id="1377232898" mimeType="audio/mp4" segmentAlignment="0"
 lang="eng">
      <Label>eng</Label>
      <SegmentTemplate timescale="48000" media="index_audio_$RepresentationID$_0_
$Number$.mp4?m=1676062374" initialization="index_audio_$RepresentationID$_0_init.mp4?
m=1676062374" startNumber="156" presentationTimeOffset="3861387246">
        <SegmentTimeline>
          <S t="3861387246" d="227328"/>
          <S t="3861614574" d="288768"/>
          <S t="3861903342" d="287744"/>
          <S t="3862191086" d="288768"/>
          <S t="3862479854" d="287744"/>
          <S t="3862767598" d="288768"/>
          <S t="3863056366" d="287744"/>
          <S t="3863344110" d="202752"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="2" bandwidth="193007" audioSamplingRate="48000"
 codecs="mp4a.40.2">
        <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
```

```
      <Representation id="4" bandwidth="193007" audioSamplingRate="48000"
 codecs="mp4a.40.2">
        <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
      <Representation id="6" bandwidth="193007" audioSamplingRate="48000"
 codecs="mp4a.40.2">
        <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
    </AdaptationSet>
    <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
 value="2023-02-10T21:07:35.111Z"/>
  </Period>
  <Period start="PT80490.538S" id="163">
    <AdaptationSet id="1485523442" mimeType="video/mp4" segmentAlignment="true"
 startWithSAP="1" subsegmentAlignment="true" subsegmentStartsWithSAP="1"
 bitstreamSwitching="true">
      <SegmentTemplate timescale="60000" media="index_video_$RepresentationID$_0_
$Number$.mp4?m=1676062374" initialization="index_video_$RepresentationID$_0_init.mp4?
m=1676062374" startNumber="164" presentationTimeOffset="4829432310">
        <SegmentTimeline>
          <S t="4829432310" d="348348"/>
          <S t="4829780658" d="360360" r="1"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="1" width="960" height="540" frameRate="30000/1001"
 bandwidth="1800000" codecs="avc1.4D401F"/>
      <Representation id="3" width="640" height="360" frameRate="30000/1001"
 bandwidth="1200000" codecs="avc1.4D401E"/>
      <Representation id="5" width="480" height="270" frameRate="30000/1001"
 bandwidth="800000" codecs="avc1.4D4015"/>
    </AdaptationSet>
    <AdaptationSet id="1377232898" mimeType="audio/mp4" segmentAlignment="0"
 lang="eng">
      <Label>eng</Label>
      <SegmentTemplate timescale="48000" media="index_audio_$RepresentationID$_0_
$Number$.mp4?m=1676062374" initialization="index_audio_$RepresentationID$_0_init.mp4?
m=1676062374" startNumber="164" presentationTimeOffset="3863546862">
        <SegmentTimeline>
          <S t="3863546862" d="278528"/>
          <S t="3863825390" d="287744"/>
          <S t="3864113134" d="288768"/>
        </SegmentTimeline>
```

```
      </SegmentTemplate>
      <Representation id="2" bandwidth="193007" audioSamplingRate="48000"
 codecs="mp4a.40.2">
        <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
      <Representation id="4" bandwidth="193007" audioSamplingRate="48000"
 codecs="mp4a.40.2">
        <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
      <Representation id="6" bandwidth="193007" audioSamplingRate="48000"
 codecs="mp4a.40.2">
        <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
    </AdaptationSet>
    <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
 value="2023-02-10T21:08:20.090Z"/>
  </Period>
</MPD>
```

Example Linear DASH personalized manifest (with creative ad signaling):

```
<MPD availabilityStartTime="2023-02-09T22:47:05.865000+00:00"
 id="201" minBufferTime="PT10S" minimumUpdatePeriod="PT6S"
 profiles="urn:mpeg:dash:profile:isoff-live:2011"
 publishTime="2023-02-10T21:08:43+00:00" suggestedPresentationDelay="PT20.000S"
 timeShiftBufferDepth="PT88.999S" type="dynamic" xmlns="urn:mpeg:dash:schema:mpd:2011"
 xmlns:scte35="urn:scte:scte35:2013:xml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/
ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd">
    <BaseURL>https://d3fch9e2fcarly.cloudfront.net/out/v1/
f9f38deca3f14fc4b5ab3cdbd76cfb9e/</BaseURL>
    <Location>https://777788889999.mediatailor.us-west-2.amazonaws.com/
v1/dash/94063eadf7d8c56e9e2edd84fdf897826a70d0df/emt/out/v1/
f9f38deca3f14fc4b5ab3cdbd76cfb9e/index.mpd?
aws.sessionId=672ed481-4ffd-4270-936f-7c8403947f2e</Location>
    <Period duration="PT304.103S" id="104" start="PT80141.456S">
        <AdaptationSet bitstreamSwitching="true" id="1485523442" mimeType="video/
mp4" segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
```

```
            <SegmentTemplate initialization="index_video_$RepresentationID$_0_init.mp4?
m=1676062374" media="index_video_$RepresentationID$_0_$Number$.mp4?m=1676062374"
 presentationTimeOffset="4808487386" startNumber="151" timescale="60000">
                <SegmentTimeline>
                    <S d="360360" r="3" t="4824975858"/>
                    <S d="316316" t="4826417298"/>
                </SegmentTimeline>
            </SegmentTemplate>
            <Representation bandwidth="1800000" codecs="avc1.4D401F"
 frameRate="30000/1001" height="540" id="1" width="960"/>
            <Representation bandwidth="1200000" codecs="avc1.4D401E"
 frameRate="30000/1001" height="360" id="3" width="640"/>
            <Representation bandwidth="800000" codecs="avc1.4D4015"
 frameRate="30000/1001" height="270" id="5" width="480"/>
        </AdaptationSet>
        <AdaptationSet id="1377232898" lang="eng" mimeType="audio/mp4"
 segmentAlignment="0">
            <Label>eng</Label>
            <SegmentTemplate initialization="index_audio_$RepresentationID$_0_init.mp4?
m=1676062374" media="index_audio_$RepresentationID$_0_$Number$.mp4?m=1676062374"
 presentationTimeOffset="3846790126" startNumber="151" timescale="48000">
                <SegmentTimeline>
                    <S d="287744" t="3859981294"/>
                    <S d="288768" t="3860269038"/>
                    <S d="287744" t="3860557806"/>
                    <S d="288768" t="3860845550"/>
                    <S d="252928" t="3861134318"/>
                </SegmentTimeline>
            </SegmentTemplate>
            <Representation audioSamplingRate="48000" bandwidth="193007"
 codecs="mp4a.40.2" id="2">
                <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            </Representation>
            <Representation audioSamplingRate="48000" bandwidth="193007"
 codecs="mp4a.40.2" id="4">
                <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            </Representation>
            <Representation audioSamplingRate="48000" bandwidth="193007"
 codecs="mp4a.40.2" id="6">
                <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            </Representation>
```

```
        </AdaptationSet>
        <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
 value="2023-02-10T21:02:31.007Z"/>
    </Period>
    <Period id="155_1" start="PT22H20M45.56S">
        <BaseURL>https://777788889999.mediatailor.us-west-2.amazonaws.com/
v1/dashsegment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/
emt/672ed481-4ffd-4270-936f-7c8403947f2e/155/155_1/</BaseURL>
        <EventStream schemeIdUri="urn:sva:advertising-wg:ad-id-signaling"
 timescale="90000">
            <Event presentationTime="xxxxx" duration="1351350">
                <![CDATA[{"version": 1,"identifiers": [{"scheme":
 "urn:smpte:ul:060E2B34.01040101.01200900.00000000","value": "155_1","ad_position":
 "155_1", "ad_type":"avail","creative_id": "123","tracking_uri": "../../../../v1/
tracking/hashed-account-id/origin-id/session-id","custom_vast_data":"123abc"}]}]]>
            </Event>
        </EventStream>
        <AdaptationSet bitstreamSwitching="false" frameRate="30000/1001"
 mimeType="video/mp4" segmentAlignment="true" startWithSAP="1"
 subsegmentAlignment="true" subsegmentStartsWithSAP="1">
            <SegmentTemplate startNumber="1" timescale="90000"/>
            <Representation bandwidth="1800000" codecs="avc1.64001f" height="540"
 id="1" width="960">
                <SegmentTemplate initialization="asset_540_1_2init.mp4"
 media="asset_540_1_2_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180180" r="6" t="0"/>
                        <S d="90090" t="1261260"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="1200000" codecs="avc1.64001e" height="360"
 id="3" width="640">
                <SegmentTemplate initialization="asset_360_1_1init.mp4"
 media="asset_360_1_1_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180180" r="6" t="0"/>
                        <S d="90090" t="1261260"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="800000" codecs="avc1.640015" height="270" id="5"
 width="480">
```

```
                <SegmentTemplate initialization="asset_270_0_0init.mp4"
 media="asset_270_0_0_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180180" r="6" t="0"/>
                        <S d="90090" t="1261260"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
        <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="asset_audio_128_3init.mp4"
 media="asset_audio_128_3_$Number%09d$.mp4" startNumber="1" timescale="48000"/>
            <Label>eng</Label>
            <Representation audioSamplingRate="48000" bandwidth="128000"
 codecs="mp4a.40.2" id="6">
                <SegmentTemplate initialization="asset_audio_128_3init.mp4"
 media="asset_audio_128_3_$Number%09d$.mp4" startNumber="1" timescale="48000">
                    <SegmentTimeline>
                        <S d="98304" t="0"/>
                        <S d="96256" r="1" t="98304"/>
                        <S d="95232" t="290816"/>
                        <S d="96256" r="2" t="386048"/>
                        <S d="48128" t="674816"/>
                    </SegmentTimeline>
                </SegmentTemplate>
                <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            </Representation>
        </AdaptationSet>
    </Period>
    <Period id="155_2" start="PT22H21M0.575S">
        <BaseURL>https://777788889999.mediatailor.us-west-2.amazonaws.com/
v1/dashsegment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/
emt/672ed481-4ffd-4270-936f-7c8403947f2e/155/155_2/</BaseURL>
        <EventStream schemeIdUri="urn:sva:advertising-wg:ad-id-signaling"
 timescale="90000">
            <Event presentationTime="0" duration="1351350">
                <![CDATA[{"version": 1,"identifiers": [{"scheme":
 "urn:smpte:ul:060E2B34.01040101.01200900.00000000","value": "155_2","ad_position":
 "155_2", "ad_type":"avail","creative_id": "234","tracking_uri": "../../../../v1/
tracking/hashed-account-id/origin-id/session-id","custom_vast_data":"123abc"}]}]]>
            </Event>
        </EventStream>
```

```
       <AdaptationSet bitstreamSwitching="false" frameRate="30000/1001"
mimeType="video/mp4" segmentAlignment="true" startWithSAP="1"
subsegmentAlignment="true" subsegmentStartsWithSAP="1">
           <SegmentTemplate startNumber="1" timescale="90000"/>
           <Representation bandwidth="1800000" codecs="avc1.64001f" height="540"
id="1" width="960">
               <SegmentTemplate initialization="asset_540_1_2init.mp4"
media="asset_540_1_2_$Number%09d$.mp4" startNumber="1" timescale="90000">
                   <SegmentTimeline>
                       <S d="180180" r="6" t="0"/>
                       <S d="90090" t="1261260"/>
                   </SegmentTimeline>
               </SegmentTemplate>
           </Representation>
           <Representation bandwidth="1200000" codecs="avc1.64001e" height="360"
id="3" width="640">
               <SegmentTemplate initialization="asset_360_1_1init.mp4"
media="asset_360_1_1_$Number%09d$.mp4" startNumber="1" timescale="90000">
                   <SegmentTimeline>
                       <S d="180180" r="6" t="0"/>
                       <S d="90090" t="1261260"/>
                   </SegmentTimeline>
               </SegmentTemplate>
           </Representation>
           <Representation bandwidth="800000" codecs="avc1.640015" height="270" id="5"
width="480">
               <SegmentTemplate initialization="asset_270_0_0init.mp4"
media="asset_270_0_0_$Number%09d$.mp4" startNumber="1" timescale="90000">
                   <SegmentTimeline>
                       <S d="180180" r="6" t="0"/>
                       <S d="90090" t="1261260"/>
                   </SegmentTimeline>
               </SegmentTemplate>
           </Representation>
       </AdaptationSet>
       <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
           <SegmentTemplate initialization="asset_audio_128_3init.mp4"
media="asset_audio_128_3_$Number%09d$.mp4" startNumber="1" timescale="48000"/>
           <Label>eng</Label>
           <Representation audioSamplingRate="48000" bandwidth="128000"
codecs="mp4a.40.2" id="6">
               <SegmentTemplate initialization="asset_audio_128_3init.mp4"
media="asset_audio_128_3_$Number%09d$.mp4" startNumber="1" timescale="48000">
                   <SegmentTimeline>
```

```
                        <S d="98304" t="0"/>
                        <S d="96256" r="1" t="98304"/>
                        <S d="95232" t="290816"/>
                        <S d="96256" r="2" t="386048"/>
                        <S d="48128" t="674816"/>
                    </SegmentTimeline>
                </SegmentTemplate>
                <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            </Representation>
        </AdaptationSet>
    </Period>
    <Period id="155_3" start="PT22H21M15.59S">
        <BaseURL>https://777788889999.mediatailor.us-west-2.amazonaws.com/
v1/dashsegment/94063eadf7d8c56e9e2edd84fdf897826a70d0df/
emt/672ed481-4ffd-4270-936f-7c8403947f2e/155/155_3/</BaseURL>
        <EventStream schemeIdUri="urn:sva:advertising-wg:ad-id-signaling"
 timescale="90000">
            <Event presentationTime="0" duration="1351350">
                <![CDATA[{"version": 1,"identifiers": [{"scheme":
 "urn:smpte:ul:060E2B34.01040101.01200900.00000000","value": "155_3","ad_position":
 "155_3", "ad_type":"avail","creative_id": "345","tracking_uri": "../../../../v1/
tracking/hashed-account-id/origin-id/session-id","custom_vast_data":"123abc"}]}]]>
            </Event>
        </EventStream>
        <AdaptationSet bitstreamSwitching="false" frameRate="30000/1001"
 mimeType="video/mp4" segmentAlignment="true" startWithSAP="1"
 subsegmentAlignment="true" subsegmentStartsWithSAP="1">
            <SegmentTemplate startNumber="1" timescale="90000"/>
            <Representation bandwidth="1800000" codecs="avc1.64001f" height="540"
 id="1" width="960">
                <SegmentTemplate initialization="asset_540_1_2init.mp4"
 media="asset_540_1_2_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180180" r="6" t="0"/>
                        <S d="90090" t="1261260"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="1200000" codecs="avc1.64001e" height="360"
 id="3" width="640">
                <SegmentTemplate initialization="asset_360_1_1init.mp4"
 media="asset_360_1_1_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
```

```
                        <S d="180180" r="6" t="0"/>
                        <S d="90090" t="1261260"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
            <Representation bandwidth="800000" codecs="avc1.640015" height="270" id="5"
 width="480">
                <SegmentTemplate initialization="asset_270_0_0init.mp4"
 media="asset_270_0_0_$Number%09d$.mp4" startNumber="1" timescale="90000">
                    <SegmentTimeline>
                        <S d="180180" r="6" t="0"/>
                        <S d="90090" t="1261260"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
        <AdaptationSet lang="eng" mimeType="audio/mp4" segmentAlignment="0">
            <SegmentTemplate initialization="asset_audio_128_3init.mp4"
 media="asset_audio_128_3_$Number%09d$.mp4" startNumber="1" timescale="48000"/>
            <Label>eng</Label>
            <Representation audioSamplingRate="48000" bandwidth="128000"
 codecs="mp4a.40.2" id="6">
                <SegmentTemplate initialization="asset_audio_128_3init.mp4"
 media="asset_audio_128_3_$Number%09d$.mp4" startNumber="1" timescale="48000">
                    <SegmentTimeline>
                        <S d="98304" t="0"/>
                        <S d="96256" r="1" t="98304"/>
                        <S d="95232" t="290816"/>
                        <S d="96256" r="2" t="386048"/>
                        <S d="48128" t="674816"/>
                    </SegmentTimeline>
                </SegmentTemplate>
                <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            </Representation>
        </AdaptationSet>
    </Period>
    <Period id="163" start="PT80490.538S">
        <AdaptationSet bitstreamSwitching="true" id="1485523442" mimeType="video/
mp4" segmentAlignment="true" startWithSAP="1" subsegmentAlignment="true"
 subsegmentStartsWithSAP="1">
            <SegmentTemplate initialization="index_video_$RepresentationID$_0_init.mp4?
m=1676062374" media="index_video_$RepresentationID$_0_$Number$.mp4?m=1676062374"
 presentationTimeOffset="4829432310" startNumber="164" timescale="60000">
```

```
                <SegmentTimeline>
                    <S d="348348" t="4829432310"/>
                    <S d="360360" r="1" t="4829780658"/>
                </SegmentTimeline>
            </SegmentTemplate>
            <Representation bandwidth="1800000" codecs="avc1.4D401F"
 frameRate="30000/1001" height="540" id="1" width="960"/>
            <Representation bandwidth="1200000" codecs="avc1.4D401E"
 frameRate="30000/1001" height="360" id="3" width="640"/>
            <Representation bandwidth="800000" codecs="avc1.4D4015"
 frameRate="30000/1001" height="270" id="5" width="480"/>
        </AdaptationSet>
        <AdaptationSet id="1377232898" lang="eng" mimeType="audio/mp4"
 segmentAlignment="0">
            <Label>eng</Label>
            <SegmentTemplate initialization="index_audio_$RepresentationID$_0_init.mp4?
m=1676062374" media="index_audio_$RepresentationID$_0_$Number$.mp4?m=1676062374"
 presentationTimeOffset="3863546862" startNumber="164" timescale="48000">
                <SegmentTimeline>
                    <S d="278528" t="3863546862"/>
                    <S d="287744" t="3863825390"/>
                    <S d="288768" t="3864113134"/>
                </SegmentTimeline>
            </SegmentTemplate>
            <Representation audioSamplingRate="48000" bandwidth="193007"
 codecs="mp4a.40.2" id="2">
                <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            </Representation>
            <Representation audioSamplingRate="48000" bandwidth="193007"
 codecs="mp4a.40.2" id="4">
                <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            </Representation>
            <Representation audioSamplingRate="48000" bandwidth="193007"
 codecs="mp4a.40.2" id="6">
                <AudioChannelConfiguration
 schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            </Representation>
        </AdaptationSet>
        <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
 value="2023-02-10T21:08:20.090Z"/>
    </Period>
```

```
</MPD>
```

Ad Decision Server (ADS) interactions

MediaTailor uses the creative `id` attribute value from the VAST response as a value in the ad ID signaling. If the `id` attribute value is empty or not present in the VAST response, MediaTailor places an empty value in the ad ID signaling.

Example VAST response:

The following sample VAST response includes an `id` attribute value for the inline linear `Creative`. MediaTailor extracts the value from the custom VAST `Extension` element and places that value in the creative metadata of the manifest.

```
<?xml version="1.0" encoding="utf-8"?>
<VAST version="3.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Ad sequence="3">
        <InLine>
            <AdSystem>2.0</AdSystem>
            <AdTitle>AD-caribbean2-15</AdTitle>
            <Impression><![CDATA[https://n8ljfs0xxx.execute-api.us-
west-2.amazonaws.com/v1/impression]]></Impression>
            <Creatives>
                <Creative sequence="3" apiFramework="inLine" id="1234">
                    <Linear>
                        <Duration>00:00:15</Duration>
                        <MediaFiles>
                            <MediaFile id="00002" delivery="progressive" type="video/
mp4" width="1280" height="720"><![CDATA[https://d3re4i3vgppxxx.cloudfront.net/Media/
Bumpers/AD-caribbean2-15-HD.mp4]]></MediaFile>
                        </MediaFiles>
                    </Linear>
                </Creative>
            </Creatives>
          <Extensions>
            <Extension type="creative_signaling"><![CDATA[999999|
TVNlDDNpFTchtpRj,E5TfTtcYd5IEzvEt,ChA05OHcvWRGFY6Zp5VSSlxUEJ2B9p8GGhQIDzIQkFeQC-
Ho67FR3P9qNa6khSAGKgAyAA]]></Extension>
          </Extensions>
        </InLine>
    </Ad>
</VAST>
```

Client-side tracking API

The following example shows how a player SDK links the ad metadata in the manifest with the full tracking event data in the client-side tracking response payload with `creativeId` and `adId`.

Example JSON message:

```
{
  "avails": [
    {
      "adBreakTrackingEvents": [],
      "ads": [
        {
          "adId": "5",
          "adParameters": "",
          "adProgramDateTime": null,
          "adSystem": "2.0",
          "adTitle": "AD-caribbean2-15",
          "adVerifications": [],
          "companionAds": [],
          "creativeId": "1234",
          "creativeSequence": "2",
          "duration": "PT15S",
          "durationInSeconds": 15,
          "extensions": [],
          "mediaFiles": {
            "mediaFilesList": [],
            "mezzanine": ""
          },
          "skipOffset": null,
          "startTime": "PT30S",
          "startTimeInSeconds": 30,
          "trackingEvents": [
            {
              "beaconUrls": [
                "https://myServer/impression"
              ],
              "duration": "PT15S",
              "durationInSeconds": 15,
              "eventId": "5",
              "eventProgramDateTime": null,
              "eventType": "impression",
              "startTime": "PT30S",
```

```
              "startTimeInSeconds": 30
            }
          ],
          "vastAdId": ""
        }
      ],
      "availId": "5",
      "availProgramDateTime": null,
      "duration": "PT15S",
      "durationInSeconds": 15,
      "meta": null,
      "nonLinearAdsList": [],
      "startTime": "PT30S",
      "startTimeInSeconds": 30
    }
  ],
  "nextToken": "UFQ1TTM0Ljk2N1NfMjAyMi0xMS0xOFQwNDozMzo1Mi4yNDUxOTdaXzE%3D",
  "nonLinearAvails": []
}
```

Using AWS Elemental MediaTailor to create linear assembled streams

AWS Elemental MediaTailor channel assembly is a manifest-only service that allows you to create linear streaming channels using your existing video on demand (VOD) content mixed with live content. MediaTailor never touches your content segments, which are served directly from your origin server. Instead, MediaTailor fetches the manifests from your origin, and uses them to assemble a live sliding manifest window that references the underlying content segments. Channel assembly keeps track of things like the media sequence number that's necessary to make playback smooth from asset to asset. Linear assembled streams are created with a low running cost by using existing multi-bitrate encoded and packaged VOD content.

You can easily monetize channel assembly linear streams by inserting ad breaks in your programs without having to condition the content with SCTE-35 markers. You can use channel assembly with the MediaTailor ad insertion service, or any server-side ad insertion service.

To get started with channel assembly, see the section called “Getting started with MediaTailor channel assembly”.

Topics

- Working with source locations
- Working with channels
- Adding a program to a channel's schedule
- Insert personalized ads and ad breaks in a channel stream
- Enable time-shifted viewing
- Troubleshooting playback errors returned by MediaTailor

Working with source locations

A source location represents the origin server where your source content is stored. A source location can be Amazon S3, a standard web server, a content delivery network (CDN) such as Amazon CloudFront, or a packaging origin such as AWS Elemental MediaPackage. MediaTailor retrieves your content manifests from the source location, and uses them to assemble your channel's linear stream.

This topic explains how to use the AWS Elemental MediaTailor console to create and delete source locations, and how to work with VOD sources.

Topics

- Creating a source location
- Configuring authentication for your source location
- Working with VOD sources
- Working with live sources
- Using package configurations
- Manifest caching

Creating a source location

The following procedure explains how to create a source location using the MediaTailor console. For information about how to create source locations using the MediaTailor API, see CreateSourceLocation in the *AWS Elemental MediaTailor API Reference*.

To create a source location

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Source locations**.
3. On the navigation bar, choose **Create source location**.
4. Under **Source location configuration**, enter a name and the base URL of your origin server:

   - **Name**: An identifier for your source location, such as **my-origin**.
   - **Base URL**: The protocol and base URL of the origin server your content is stored, such as **https://111111111111.cloudfront.net**. The URL must be in a standard HTTP URL format, prefixed with **http://** or **https://**.

     Optionally select **Use SigV4 for Amazon S3 authentication** if your source location is an Amazon S3 bucket, and if you'd like to use AWS Signature Version 4 for Amazon S3 access authentication. For advanced information, see Configuring authentication for your source location.

5. Under **Access configuration**, optionally configure authentication for your source location:

- **Access type**: Select the authentication type that MediaTailor uses to access the content stored on the source location's origin.
  - **SigV4 for Amazon S3** - MediaTailor uses Amazon Signature Version 4 (SigV4) to authorize request to your origin. For more information, see the section called "Authenticating requests to Amazon S3 with SigV4".
  - **Secrets Manager access token authentication** - MediaTailor uses Secrets Manager and a AWS KMS customer managed key created, owned, and managed by you to facilitate access token authentication between MediaTailor and your origin. For information about how to configure **Secrets Manager access token authentication**, see the section called "Working with AWS Secrets Manager access token authentication".
    - **Header name** - Specify a HTTP header name. MediaTailor uses the HTTP header to send the access token to your origin in content manifest requests. You can can use any header name as long as it doesn't start with `x-amz-` or `x-amzn-`. If you're integrating with MediaPackage CDN authorization, the header value should be `X-MediaPackage-CDNIdentifier`.
    - **Secret string key** - The `SecretString` key that you specified in your Secrets Manager secret. For example, if your `SecretString` contains a key and value pair such as: `{"MyHeaderName": "11111111-2222-3333-4444-111122223333"}`, then `MyHeaderName` is the `SecretString` key you enter in this field.
    - **Secret ARN** - The ARN of the secret that holds your access token. For a step-by-step guide, see Step 2: Create an AWS Secrets Manager secret.

6. Under **Segment delivery server configuration**, optionally configure a server to deliver your content segments:

- **Use a default segment delivery server**: Enter the base URL of the server that is used to deliver your content segments, such as a CDN. Configure **Default segment host name** if you'd like to use a different server than the source location server to serve the content segments. For example, you can restrict access to the origin manifests from players by using a different CDN configuration for the **Base HTTP URL** (what MediaTailor uses to access the manifests) and the **Default Segment Base URL** (what players uses to access the content segments). If you don't enter a value, MediaTailor defaults to the source location server for segment delivery.
- **Use named segment delivery servers**: If you have configured a default segment delivery server, you can also configure additional segment delivery servers. Each one must have a unique name and a base URL. The base URL can be a full HTTP URL, or it can be a relative

path like `/some/path/` . The names are used to identify which server should be used when MediaTailor receives a request for content segments. If the request contains the header `X-MediaTailor-SegmentDeliveryConfigurationName` and the value of the header matches a name, the corresponding base URL will be used to serve the content. If the header is not included in the request, or if it does not match any names, then the default segment delivery server will be used.

7. Choose **Create source location**.
8. To add more source locations, repeat steps 2-6.

Configuring authentication for your source location

Use **access configuration** to configure authentication for your source location. When access configuration is on, MediaTailor only retrieves source manifests from your origin if the request is authorized between MediaTailor and your origin. Access configuration is turned off by default.

MediaTailor supports the following authentication types:

- SigV4 for Amazon S3 authentication
- AWS Secrets Manager access token
- SigV4 for MediaPackage version 2 (v2) authentication

This chapter explains how to use SigV4 for Amazon S3, MediaPackage v2, and AWS Secrets Manager access tokens for source location authentication.

For more information, select the applicable topic.

Topics

- Authenticating requests to Amazon S3 with SigV4
- Working with SigV4 for MediaPackage Version 2
- Working with AWS Secrets Manager access token authentication

Authenticating requests to Amazon S3 with SigV4

Signature Version 4 (SigV4) for Amazon S3 is a signing protocol used to authenticate requests to Amazon S3 over HTTPS. When you use SigV4 for Amazon S3, MediaTailor includes a signed

authorization header in the HTTPS request to the Amazon S3 bucket used as your origin. If the signed authorization header is valid, your origin fulfills the request. If it isn't valid, the request fails.

For general information about SigV4 for AWS Key Management Service, see the Authenticating Requests (AWS Signature Version 4) topic in the *Amazon S3 API reference*.

> **Note**
>
> MediaTailor always signs requests to these origins with SigV4.

Requirements

If you activate SigV4 for Amazon S3 authentication for your source location, you must meet these requirements:

- You must allow MediaTailor to access your Amazon S3 bucket by granting **mediatailor.amazonaws.com** principal access in IAM. For information about configuring access in IAM, see Access management in the *AWS Identity and Access Management User Guide*.
- The **mediatailor.amazonaws.com** service principal must have permissions to read all top-level manifests referenced by the VOD source package configurations.
- The caller of the API must have **s3:GetObject** IAM permissions to read all top-level manifests referenced by your MediaTailor VOD source package configurations.
- Your MediaTailor source location base URL must follow the Amazon S3 virtual hosted-style request URL format. For example, https://*bucket-name*.s3.*Region*.amazonaws.com/*key-name*. For information about Amazon S3 hosted virtual-style access, see Virtual Hosted-Style Requests.

Working with SigV4 for MediaPackage Version 2

Signature Version 4 (SigV4) for MediaPackage v2 is a signing protocol used to authenticate requests to MediaPackage v2 over HTTP. When you use SigV4 for MediaPackage v2, MediaTailor includes a signed authorization header in the HTTP request to the MediaPackage v2 endpoint used as your origin. If the signed authorization header is valid, your origin fulfills the request. If it isn't valid, the request fails.

For general information about SigV4 for MediaPackage v2, see the Authenticating Requests (AWS Signature Version 4) topic in the *MediaPackage v2 API reference*.

Requirements

If you activate SigV4 for MediaPackage v2 authentication for your source location, you must meet these requirements:

- You must allow MediaTailor to access your MediaPackage v2 endpoint by granting **mediatailor.amazonaws.com** principal access in an Origin Access Policy on the endpoint.
- Your MediaTailor source location base URL must be a MediaPackage v2 endpoint.
- The caller of the API must have **mediapackagev2:GetObject** IAM permissions to read all top-level manifests referenced by the MediaTailor source packaging configurations.

Working with AWS Secrets Manager access token authentication

MediaTailor supports *Secrets Manager access token authentication*. With AWS Secrets Manager access token authentication, MediaTailor uses an AWS Key Management Service (AWS KMS) customer managed key and an AWS Secrets Manager secret that you create, own, and manage to authenticate requests to your origin.

In this section, we explain how Secrets Manager access token authentication works, and provide step-by-step information about how to configure Secrets Manager access token authentication. You can work with Secrets Manager access token authentication in the AWS Management Console or programmatically with AWS APIs.

Topics

- Configuring AWS Secrets Manager access token authentication
- Integrating with MediaPackage endpoints that use CDN authorization
- How MediaTailor Secrets Manager access token authentication works

Configuring AWS Secrets Manager access token authentication

When you want to use AWS Secrets Manager access token authentication, you perform the following steps:

1. You create an AWS Key Management Service customer managed key.
2. You create a AWS Secrets Manager secret. The secret contains your access token, which is stored in Secrets Manager as an encrypted secret value. MediaTailor uses the AWS KMS customer managed key to decrypt the secret value.

3. You configure an AWS Elemental MediaTailor source location to use Secrets Manager access token authentication.

The following section provides step-by-step guidance on how to configure AWS Secrets Manager access token authentication.

Topics

- Step 1: Create an AWS KMS symmetric customer managed key
- Step 2: Create an AWS Secrets Manager secret
- Step 3: Configure a MediaTailor source location with access token authentication

Step 1: Create an AWS KMS symmetric customer managed key

You use AWS Secrets Manager to store your access token in the form of a `SecretString` stored in a secret. The `SecretString` is encrypted through the use of an *AWS KMS symmetric customer managed key* that you create, own, and manage. MediaTailor uses the symmetric customer managed key to facilitate access to the secret with a grant, and to encrypt and decrypt the secret value.

Customer managed keys let you perform tasks such as the following:

- Establishing and maintaining key policies
- Establishing and maintaining IAM policies and grants
- Enabling and disabling key policies
- Rotating cryptographic key material
- Adding tags

For information about how Secrets Manager uses AWS KMS to protect secrets, see the topic How AWS Secrets Manager uses AWS KMS in the *AWS Key Management Service Developer Guide*.

For more information about customer managed keys, see Customer managed keys in the *AWS Key Management Service Developer Guide*.

> **Note**
>
> AWS KMS charges apply for using a customer managed key For more information about pricing, see the AWS Key Management Service Pricing page.

You can create an AWS KMS symmetric customer managed key using the AWS Management Console or programmatically with the AWS KMS APIs.

To create a symmetric customer managed key

Follow the steps for Creating a symmetric customer managed key in the *AWS Key Management Service Developer Guide.*

Make a note of the key Amazon Resource Name (ARN); you'll need it in Step 2: Create an AWS Secrets Manager secret.

Encryption context

An *encryption context* is an optional set of key-value pairs that contain additional contextual information about the data.

Secrets Manager includes an encryption context when encrypting and decrypting the `SecretString`. The encryption context includes the secret ARN, which limits the encryption to that specific secret. As an added measure of security, MediaTailor creates an AWS KMS grant on your behalf. MediaTailor applies a GrantConstraints operation that only allows us to *decrypt* the `SecretString` associated with the secret ARN contained in the Secrets Manager encryption context.

For information about how Secrets Manager uses encryption context, see the Encryption context topic in the *AWS Key Management Service Developer Guide*.

Setting the key policy

Key policies control access to your customer managed key. Every customer managed key must have exactly one key policy, which contains statements that determine who can use the key and how they can use it. When you create your customer managed key you can use the default key policy. For more information, see Authentication and access control for AWS KMS in the *AWS Key Management Service Developer Guide*.

To use your customer managed key with your MediaTailor source location resources, you must give permission to the IAM principal that calls CreateSourceLocation or UpdateSourceLocation to use the following API operations:

- `kms:CreateGrant` – Adds a grant to a customer managed key. MediaTailor creates a grant on your customer managed key that lets it use the key to create or update a source location configured with access token authentication. For more information about using Grants in AWS KMS, see the *AWS Key Management Service Developer Guide.*

  This allows MediaTailor to do the following:

  - Call `Decrypt` so that it can successfully retrieve your Secrets Manager secret when calling GetSecretValue.
  - Call `RetireGrant` to retire the grant when the source location is deleted, or when access to the secret has been revoked.

The following is an example policy statement that you can add for MediaTailor:

```
{
        "Sid": "Enable MediaTailor Channel Assembly access token usage for the
 MediaTailorManagement IAM role",
        "Effect": "Allow",
        "Principal": {
        "AWS": "arn:aws:iam::account number:role/MediaTailorManagement"
    },
     "Action": "kms:CreateGrant",
     "Resource": "*",
     "Condition": {
         "StringEquals": {
             "kms:ViaService": "mediatailor.region.amazonaws.com"
         }
     }
}
```

For more information about specifying permissions in a policy and troubleshooting key access, see Grants in AWS KMS in the *AWS Key Management Service Developer Guide*.

Step 2: Create an AWS Secrets Manager secret

Use Secrets Manager to store your access token in the form of a `SecretString` that's encrypted by an AWS KMS customer managed key. MediaTailor uses the key to decrypt the `SecretString`.

For information about how Secrets Manager uses AWS KMS to protect secrets, see the topic How AWS Secrets Manager uses AWS KMS in the *AWS Key Management Service Developer Guide*.

If you use AWS Elemental MediaPackage as your source location origin, and would like to use MediaTailor Secrets Manager access token authentication follow the procedure the section called "Integrating with MediaPackage endpoints that use CDN authorization".

You can create a Secrets Manager secret using the AWS Management Console or programmatically with the Secrets Manager APIs.

To create a secret

Follow the steps for Create and manage secrets with AWS Secrets Manager in the *AWS Secrets Manager User Guide*.

Keep in mind the following considerations when creating your secret:

- The KmsKeyId must be the key ARN of the customer managed key you created in Step 1.
- You must supply a SecretString. The `SecretString` should be a valid JSON object that includes a key and value containing the access token. For example, {"MyAccessTokenIdentifier":"112233445566"}. The value must between 8-128 characters long.

  When you configure your source location with access token authentication, you specify the `SecretString` key. MediaTailor uses the key to look up and retrieve the access token stored in the `SecretString`.

  Make a note of the secret ARN and the `SecretString` key. You'll use them when you configure your source location to use access token authentication.

Attaching a resource-based secret policy

To let MediaTailor access the secret value, you must attach a resource-based policy to the secret. For more information, see Attach a permissions policy to an AWS Secrets Manager Secret in the *AWS Secrets Manager User Guide*.

The following is a policy statement example that you can add for MediaTailor:

```
{

    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "mediatailor.amazonaws.com"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "<secret ARN"
        }
    ]

}
```

Step 3: Configure a MediaTailor source location with access token authentication

You can configure Secrets Manager access token authentication using the AWS Management Console or programmatically with the MediaTailor APIs.

To configure a source location with Secrets Manager access token authentication

Follow the steps for Access configuration in the *AWS Elemental MediaTailor User Guide*.

Integrating with MediaPackage endpoints that use CDN authorization

If you use AWS Elemental MediaPackage as your source location origin, MediaTailor can integrate with MediaPackage endpoints that use CDN authorization.

To integrate with a MediaPackage endpoint that uses CDN authorization, use the following procedure.

To integrate with MediaPackage

1. Complete the steps in Setting up CDN authorization in the *AWS Elemental MediaPackage User Guide*, if you haven't already.
2. Complete the procedure in the section called “Step 1: Create an AWS KMS symmetric customer managed key”.
3. Modify the secret that you created when you set up MediaPackage CDN authorization. Modify the secret with the following values:

    - Update the `KmsKeyId` with the customer managed key ARN that you created in the section called “Step 1: Create an AWS KMS symmetric customer managed key”.

- (Optional) For the `SecretString`, you can either rotate the UUID to a new value, or you can use the existing encrypted secret as long as it's a key and value pair in a standard JSON format, such as `{"MediaPackageCDNIdentifier": "122334455667788 99"}`.

4. Complete the steps in the section called "Attaching a resource-based secret policy".
5. Complete the steps in the section called "Step 3: Configure a MediaTailor source location with access token authentication".

How MediaTailor Secrets Manager access token authentication works

After you create or update a source location to use access token authentication, MediaTailor includes the access token in an HTTP header when requesting source content manifests from your origin.

Here's an overview of how MediaTailor uses Secrets Manager access token authentication for source location origin authentication:

1. When you create or update a MediaTailor source location that uses access token authentication, MediaTailor sends a DescribeSecret request to Secrets Manager to determine the AWS KMS key associated with the secret. You include the secret ARN in your source location access configuration.
2. MediaTailor creates a grant for the customer managed key, so that MediaTailor can use the key to access and decrypt the access token stored in the SecretString. The grant name will be `MediaTailor-SourceLocation-`*`your AWS account ID`*`-`*`source location name`*.

   You can revoke access to the grant, or remove MediaTailor's access to the customer managed key at any time. For more information, see RevokeGrant in the *AWS Key Management Service API Reference*.
3. When a VOD source is created or updated, or used in a program, MediaTailor makes HTTP requests to the source locations to retrieve the source content manifests associated with the VOD sources in the source location. If the VOD source is associated with a source location that has an access token configured, the requests include the access token as an HTTP header value.

Working with VOD sources

A VOD source represents a single piece of content, such as a video or an episode of a podcast, that you add to your source location. You add one or more VOD sources to your source location, and then associate each VOD source with a program after you create your channel.

Each VOD source must have at least one *package configuration*. A package configuration specifies a package format, manifest location, and source group for your VOD source. When you create your channel, you use the package configuration's source groups to create the corresponding outputs on your channel. For example, if your source is packaged in two different formats—HLS and DASH—then you'd create two package configurations, one for DASH and one for HLS. Then, you would create two channel outputs, one for each package configuration. Each channel output provides an endpoint that's used for playback requests. So, using the preceding example, the channel would provide an endpoint for HLS playback requests and an endpoint for DASH playback requests.

If you would like the offsets of ad markers in your manifest to be detected automatically, each ad marker must appear at the same offset across all package configurations and have a duration of zero. For HLS, MediaTailor will detect `DATERANGE` and `EXT-X-CUE-OUT` tags. For DASH, HLS will detect the first Event tag within each `EventStream` tag.

In the following example, an ad break opportunity will be detected at an offset of 12000ms because of the `DATERANGE` tag with a duration of 0.0. The first `DATERANGE` tag at an offset of 0ms will not be detected because it has a duration of 10.0.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-DATERANGE:ID="1001",START-DATE="2021-09-16T23:51:05.249Z",DURATION=10.0,SCTE35-
OUT=0xFC3025000000003289800FFF01405000003E97FEFFE1D381BD8FE000DBBA00001010100000FD2B275
#EXTINF:6.000,
../../../719f911124e0495cbb067c91c1d6c298/1785a16ca14d4c2884781f25333f6766/index_1_0.ts
#EXTINF:6.000,
../../../719f911124e0495cbb067c91c1d6c298/1785a16ca14d4c2884781f25333f6766/index_1_1.ts
#EXT-X-DATERANGE:ID="1001",START-DATE="2021-09-16T23:51:05.249Z",DURATION=0.0,SCTE35-
OUT=0xFC3025000000003289800FFF01405000003E97FEFFE1D381BD8FE000DBBA00001010100000FD2B275
#EXTINF:6.000,
../../../719f911124e0495cbb067c91c1d6c298/1785a16ca14d4c2884781f25333f6766/index_1_2.ts
```

In the following example, an ad break opportunity will be detected at an offset of 0ms because the EXT-X-CUE-OUT tag has a duration of 0 and is followed immediately by an EXT-X-CUE-IN tag. The second EXT-X-CUE-OUT/EXT-X-CUE-IN pair will not be detected because it has a duration of 10.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-CUE-OUT:0
#EXT-X-CUE-IN
#EXTINF:6.000,
../../../719f911124e0495cbb067c91c1d6c298/1785a16ca14d4c2884781f25333f6766/index_1_0.ts
#EXTINF:6.000,
../../../719f911124e0495cbb067c91c1d6c298/1785a16ca14d4c2884781f25333f6766/index_1_1.ts
#EXT-X-CUE-OUT:10
...
#EXT-X-CUE-IN
#EXTINF:6.000,
../../../719f911124e0495cbb067c91c1d6c298/1785a16ca14d4c2884781f25333f6766/index_1_2.ts
```

In the following example, an ad break opportunity will be detected at an offset of 0ms because the first Event in the EventStream occurs in the period starting at PT0.000S. The second Event in the EventStream will not be detected.

```
<Period start="PT0.000S" id="9912561" duration="PT29.433S">
<EventStream timescale="90000" schemeIdUri="urn:scte:scte35:2013:xml">
<Event duration="0">
  <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="1241950593" tier="4095">
    <scte35:SpliceInsert spliceEventId="99" spliceEventCancelIndicator="false"
 outOfNetworkIndicator="true" spliceImmediateFlag="false" uniqueProgramId="1"
 availNum="1" availsExpected="1">
      <scte35:Program><scte35:SpliceTime ptsTime="3552273000"/></scte35:Program>
      <scte35:BreakDuration autoReturn="true" duration="2700000"/>
    </scte35:SpliceInsert>
  </scte35:SpliceInfoSection>
</Event>
<Event duration="0">
  <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="1241950593" tier="4095">
    <scte35:SpliceInsert spliceEventId="99" spliceEventCancelIndicator="false"
 outOfNetworkIndicator="true" spliceImmediateFlag="false" uniqueProgramId="1"
 availNum="1" availsExpected="1">
```

```
      <scte35:Program><scte35:SpliceTime ptsTime="3552273000"/></scte35:Program>
      <scte35:BreakDuration autoReturn="true" duration="2700000"/>
    </scte35:SpliceInsert>
  </scte35:SpliceInfoSection>
</Event>
</EventStream>
  ...
</Period>
```

Adding VOD sources to your source location

The following procedure explains how to add VOD sources to your source location and set up package configurations using the MediaTailor console. For information about how to add VOD sources using the MediaTailor API, see CreateVodSource in the *AWS Elemental MediaTailor API Reference*.

Important

Before you add your VOD sources, make sure that they meet these requirements:

- Source variants must all have the same length, as determined by the source manifest.
- Within a package configuration, each source must have the same number of child streams.

Because of these requirements, we don't support per title or automated ABR, because these encoding methods can produce varying manifest lengths and child streams.
We recommend that you use an encoding template that includes a minimum segment length to ensure that your encoded sources meet these requirements.

To add VOD sources to your source locations

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Source locations**.
3. In the **Source locations** pane, choose the source location that you created in the To create a source location procedure.
4. Choose **Add VOD source**.
5. Under **VOD source details**, enter a name for your VOD source:

- **Name**: An identifier for your VOD source, such as **my-example-video**.

6. Under **Package configurations** > *source-group-name* enter information about the package configuration:

> **Note**
>
> Your source's package configurations must all have the same duration, as determined by the source's manifest. And, all of the sources within a package configuration must have the same number of child streams. To meet these requirements, we recommend that you use an encoding template for your assets. We recommend that you use an encoding template with a minimum segment length of one second. MediaTailor doesn't support per title or automated adaptive bitrate streaming (ABR) because these encoding methods violate these requirements.

- **Source group**: Enter a source group name that describes this package configuration, such as HLS-4k. Make a note of this name; you'll reference it when you create your channel's output. For more information, see Using source groups with your channel's outputs.
- **Type**: Select the packaged format for this configuration. MediaTailor supports HLS and DASH.
- **Relative path**: The relative path from the source location's **Base HTTP URL** to the manifest. For example, **/my/path/index.m3u8**.

> **Note**
>
> MediaTailor automatically imports all of the closed captions and child streams contained within a parent manifest. You don't need to create separate package configurations for each of your sources renditions (DASH) or variant streams (HLS).

For more information about package configurations, see Using package configurations.

7. Choose **Add VOD source**.

If you want to add more VOD sources, repeat steps 4-7 in the procedure.

Working with live sources

A *live source* represents a single live stream, such as a live football game or news broadcast, that you add to your source location. After you create your channel, you add one or more live sources to your source location, and then associate each live source with a program.

MediaTailor supports these types of linear channel assembly:

- VOD sources for a channel that contains VOD-to-live content
- Live sources for a channel that contains live-to-live content intermixed with VOD-to-live content

An example of VOD-to-live content is a channel that assembles a library of VOD assets into a live stream. One example of live-to-live content mixed with VOD-to-live content is a channel that shows mostly VOD content, except for a nightly news event or a pre-scheduled live sporting event. Another example of live-to-live content mixed with VOD-to-live content is an all live-to-live channel with origins that vary based on the time of day.

You can use live sources to set up a regional channel that shows mostly national programming, but also includes regioinal programming overrides, and has VOD content mixed in. To do so, you run one encoder/packager pair for the national content, then run regional encoders when those regions are live. Then, you create regional channel-assembly channels, each with their own schedules. This way, viewers can switch back and forth as needed. This setup helps you minimize encoding/packaging costs.

Each live source must have at least one package configuration. A *package configuration* specifies a package format, manifest location, and source group for your live source. When you create your channel, you use the package configuration's source groups to create the corresponding outputs on your channel. For example, if your source is packaged in two different formats—HLS and DASH—then you'd create two package configurations, one for DASH and one for HLS. Then, you'd create two channel outputs, one for each package configuration. Each channel output provides an endpoint that's used for playback requests. In this example, the channel provides an endpoint for HLS playback requests and an endpoint for DASH playback requests.

General requirements for using live sources

When you use live sources, your content must align with the following general requirements:

- HLS live sources - You must provide `#EXT-X-PROGRAM-DATE-TIME` tags for the first segment in the manifest window, and on every discontinuity.

- HLS - You must configure ad markers as DATERANGE.
- Source manifest window - We recommend using a manifest window with a duration that's at least as long as the manifest window on your MediaTailor Channel Assembly channel. As a best practice, consider using a manifest window duration that's 30 seconds or longer than the manifest window on the Channel Assembly channel.
- Make the target duration match the duration of the existing sources.
- Make the number of child playlists match that of the existing sources.

Configurations

If you use other AWS Elemental media services as part of your live sources workflow, we recommend following best practices when setting up your MediaPackage configuration. The following table describes how to configure MediaPackage settings based on the streaming standard you use.

MediaPackage setup for live sources

Standard	Setting	Value	Necessity	Notes
HLS	Endpoint type	`Apple HLS`	Required unless using CMAF	To match HLS `ts` AWS Elemental MediaConvert jobs
HLS	Endpoint type	`CMAF`	Required unless using Apple HLS	To match HLS `mp4` AWS Elemental MediaConvert jobs
HLS	`ProgramDateTimeIntervalSeconds`	`1`	Required	You must specify `#EXT-X-PROGRAM-DATE-TIME` on every segment in order to prevent

Standard	Setting	Value	Necessity	Notes
				playback issues when there are discontinuities.
HLS	`PlaylistWindowSeconds`	30 seconds longer than the channel assembly manifest window	Required	
HLS	`AdMarkers`	`DATERANGE`	Required when passing through ad markers	
HLS	`IncludeIframeOnlyStream`	Disabled	Recommended	
DASH	`ManifestLayout`	`FULL`	Recommended	
DASH	`SegmentTemplateFormat`	`NUMBER_WITH_TIMELINE` or `TIME_WITH_TIMELINE`	Recommended	`NUMBER_WITH_DURATION` is not supported.
DASH	`ManifestWindowSeconds`	30 seconds longer than the channel assembly manifest window	Required	

Standard	Setting	Value	Necessity	Notes
DASH	`PeriodTriggers`	ADS	Required when passing through ad markers	

Adding live sources to your source location

The following procedure explains how to use the MediaTailor console to add live sources to your source location and set up package configurations. For information about how to add live sources using the MediaTailor API, see CreateLiveSource in the *AWS Elemental MediaTailor API Reference*.

> **Important**
>
> Before you add your live sources, make sure that within a package configuration, each source has the same number of child streams.

To add live sources to your source locations

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Source locations**.
3. In the **Source locations** pane, choose the source location that you created in the To create a source location procedure.
4. On the **Live sources** tab, choose **Add live source**.
5. Under **live source details**, enter a name for your live source:
   - **Name**: An identifier for your live source, such as **my-example-video**.
6. Under **Package configurations** > *`source-group-name`* enter information about the package configuration:

   > **Note**
   >
   > Within a package configuration, all of the VOD sources and live sources must have the same number of child streams. We recommend that you configure your source streams the same way.

- **Source group**: Enter a source group name that describes this package configuration, such as HLS-4k. Make a note of this name; you'll reference it when you create your channel's output. For more information, see Using source groups with your channel's outputs.
- **Type**: Select the packaged format for this configuration. MediaTailor supports HLS and DASH.
- **Relative path**: The relative path from the source location's **Base HTTP URL** to the manifest. For example, **/my/path/index.m3u8**.

> **Note**
>
> MediaTailor automatically imports all of the closed captions and child streams contained within a parent manifest. You don't need to create separate package configurations for each of your sources renditions (DASH) or variant streams (HLS).

For more information about package configurations, see Using package configurations.

7. Choose **Add live source**.

   If you want to add more live sources, repeat steps 4-6 in the procedure.

Using package configurations

A package configuration is a representation of the source that contains the various packaging characteristics required for playback on different devices. For example, you might have a source that has three packaged formats: HLS with DRM, DASH with segment timeline addressing, and HLS with CMAF segments.

Channel assembly doesn't repackage your sources. If you want to include multiple packaged formats for a given source, you must make each packaged format available at the source location and specify the path to each packaged format.

Each package configuration object must include the following:

- **Relative path** - The full path to the source's packaged format, relative to the source location. For example, **/my/path/index.m3u8**.

- **Source group** - The name of the source group used to associate package configurations with a channel's output.
- **Type** - Either HLS or DASH.

After you have created a channel, you must also declare each source group that you want to use for the channel's output.

Manifest caching

MediaTailor periodically and opportunistically caches source playlists to improve channel assembly performance and reliability. Sometimes, the cached version becomes stale compared to the origin version at your source location. To force MediaTailor to refresh the cached version of the source, call UpdateVodSource. For example, use this call when the embedded paths change in your source. Make sure that you always keep an up-to-date version of the source available on your source location, even if you see few requests from MediaTailor.

Working with channels

A channel assembles your source manifests into a linear stream. Each channel contains one or more outputs that correspond to your package configurations.

First you create a channel, then you add your VOD sources and live sources to the channel's schedule by creating *programs*. Each program is associated with a VOD source or a live source.

Topics

- Create a channel using the MediaTailor console
- Using source groups with your channel's outputs
- Delete a channel using the MediaTailor console

Create a channel using the MediaTailor console

The following procedure describes how to create a channel using the MediaTailor console.

To create a channel

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.

3. On the navigation bar, choose **Create channel**.
4. Under **Channel details**, enter details about your channel:

   - **Name**: Enter a name for your channel.
   - **Tier**: The tier determines what features the channel supports and how much it costs to run the channel. For more information about pricing, see the Channel Assembly pricing page. MediaTailor supports the following tiers:
     - **Basic** - The Basic tier supports both the Linear and Loop playback modes, and does not support live sources.
     - **Standard** - The Standard tier supports live sources, and requires the Linear playback mode.

       When you select **Standard** in **Channel details**, you can define the audiences under **Audiences details**. These audiences will be used for programRules when you are going to create audienceMedia for your default program..

       - Choose **Add**.
       - Enter the **Audience** name in the text box. It must be between 1 and 32 alphanumeric characters long.
       - Choose **Confirm**.
       - Choose **Next**.
   - **Playback mode**: The playback mode sets the channel's playback behavior. MediaTailor supports the following playback modes:
     - **Loop** - The programs in the schedule play back-to-back in an endless loop. After the last program plays in a schedule, playback loops back to the first program. Playback continues looping until you stop the channel.
     - **Linear** - Each program in the schedule plays once, back-to-back.
5. For **Filler slate**, select the **Source location name** referencing the slate location, and the **VOD source name** to use as slate. MediaTailor uses the slate to fill gaps between programs in the schedule. If the duration of the slate is less than the duration of the gap between programs, MediaTailor loops the slate. You must configure filler slate field is if your channel uses the linear playback mode. MediaTailor doesn't support filler slate for the loop playback mode.
6. Choose **Next**.
7. Specify audience details under program rules.

8. When you select **Standard** in **Channel details**, you can define the audiences under **Audiences** details. These audiences will be used for **programRules** when you are going to create **audienceMedia** for your default program:

   - Select **Add**, and then add an Audience in the text box, then select **Confirm**.

     > **Note**
     >
     > Enter a name that's no longer than_ 32 alphanumeric characters.

   - **Output type**: Select the streaming format for the channel. DASH and HLS are supported.
   - **Source group**: Enter the name of the source group that you created in your package configuration, as described in Adding VOD sources to your source location.

9. Select **Next**.
10. Under **Manifest settings**, enter additional information about your manifest settings:

    - **Manifest window (sec)**: The time window,in seconds, contained in each manifest. The minimum value is 30 seconds, and the maximum value is 3600 seconds.
    - **Ad markup type(HLS outputs only)**: The type of ad tags that appear in VOD program ad breaks. Select `Daterange` to have MediaTailor insert ad breaks into VOD programs with `EXT-X-DATERANGE` tags. Select `Scte35 Enhanced` to have MediaTailor insert ad breaks into VOD programs using `EXT-X-CUE-OUT` and `EXT-X-CUE-IN` tags. For more information about these tag types, see SCTE-35 messages for ad breaks. For live workflows, MediaTailor always passes through `DATERANGE` tags, and doesn't pass through any Enhanced Scte35 tags, regardless of the selected Ad markup type.

11. If you want to configure multiple channel outputs, under **Outputs** choose **Add**. Then, configure the details for your output by completing the steps 6 and 7 in this procedure.
12. Choose **Next**.
13. Under **Channel policy**, choose your channel's IAM policy settings:

    - **Do not attach channel policy**: Restrict playback to only those who have access to this account's credentials.
    - **Attach custom policy**: Define your own policy and restrict access to as few or as many as you want.
    - **Attach public policy**: Accept all incoming client requests to a channel's output. You must use this option if you want to use MediaTailor ad insertion.

14. Choose **Next**.
15. Review your settings on the **Review and create** pane.
16. Choose **Create channel**.

> **Note**
>
> Channels are created in a stopped state. Your channel won't be active until you start it with the MediaTailor console or the MediaTailor StartChannel API.

Using source groups with your channel's outputs

A source group associates a package configuration with an output on a channel. When you create the package configuration on the source, you identify the source group's name. Then, when you create the output on the channel, you enter that same name to associate the output with the package configuration. VOD sources and live sources that are added to a program on a channel must belong to the source group that's identified in the output.

For example:

- VOD sources 1 and 2 both have three package configurations that have the source groups: **HLS**, **DASH**, and **HLS-4k**.
- VOD source 3 has two package configurations with source groups **HLS** and **DASH**.

If channel A has two outputs with source groups **HLS** and **DASH**, the channel output can use all three VOD sources. That's because VOD sources 1, 2, and 3 all have package configurations with source group labels **HLS** and **DASH**.

If channel B has two outputs with source groups **HLS** and **HLS-4k**, it can use VOD source 1 and 2, but not 3. This is because VOD sources 1 and 2 both have package configurations with source group labels **HLS** and **HLS-4k**.

If channel C has a single output with the source group **DASH**, it can use all three VOD sources. All three VOD sources have package configurations with the **DASH** source group.

Delete a channel using the MediaTailor console

To delete your channel, complete the following procedure.

To delete your channel

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.
3. Select the channel that you want to delete.
4. If your channel is running, from the **Actions** drop-down menu, choose **Stop**. You must stop your channel before you can delete it.
5. When your channel is stopped, from the **Actions** drop-down menu, choose **Delete**.

Adding a program to a channel's schedule

Each program contains a VOD source or a live source that's part of a source location in your account. You add your programs to your channel's schedule to control the order that they play in your channel's stream.

A program that contains a VOD source can be configured with one or more ad breaks. Each ad break contains a slate, which is a VOD source from a source location. To create the ad break, you add the slate at an offset in milliseconds into the program.

Topics

- Creating a program within a channel schedule using the MediaTailor console
- Defining audience cohorts and alternate content with Program Rules
- Generating audience-specific manifests

Creating a program within a channel schedule using the MediaTailor console

The following procedure describes how to create a program within your channel's schedule using the MediaTailor console. It also describes how to configure ad breaks, which are optional. For information about how to create programs using the MediaTailor API, see CreateProgram in the *AWS Elemental MediaTailor API Reference*.

To add a program

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.
3. In the **Channels** pane, choose the channel that you created in the To create a channel procedure.
4. In the **Program details** enter details about your program:

   - **Name**: This is the name of the program that you add to your channel.
   - **Source type**: Determines what type of source video the program plays. This option is only available for Standard channels.
     - **VOD** - The program plays a video-on-demand source, such as a pre-recorded TV episode.
     - **Live** - The program plays a live source, such as a live news broadcast.
   - **Source location name**: The source location that MediaTailor associates with the program.
     - If you choose **Select an existing source location**, choose a sourcelocation name from the **Select a source location** menu. Alternatively, search for your source location by name. This is helpful if you have a large number of source locations.
     - If you choose **Enter the source location name**, search for your source location by name.
     - **VOD source name**: The name of the VOD source that MediaTailor associates with the program:
       - If you choose **Select an existing VOD source**, select a VOD source name from the list of VOD sources that are associated with your account. Alternatively, search for your VOD source by name. This is helpful if you have a large number of VOD sources.
       - If you choose **Search by name**, search for your live source by name.
   - **Live source name**: The name of the live source to be associated with the program. This option is only available if you selected **Live** as the source type.
     - If you choose **Select an existing source location**, choose a sourcelocation name from the **Select a source location** menu. Alternatively, search for your source location by name. This is helpful if you have a large number of source locations.
     - If you choose **Enter the source location name**, search for your source location by name.
     - **VOD source name**: The name of the VOD source that MediaTailor associates with the program:

- If you choose **Select an existing live source**,select a live source name from the list of live sources that are associated with your account. You can alternatively search for your live source by name. This is helpful if you have a large number of live sources.
- If you choose **Search by name**, search for your live source by name.

5. Select **Next** to go to the **Schedule Configuration** tab.
6. Under **Playback configuration**, define when a program plays in your channel's schedule:

- **Duration in milliseconds**: Defines the duration of the program in milliseconds. This option is only available for programs that use live sources.
- **Transition type**: Defines the transitions from program to program in the schedule:
  - **Relative**: The program plays either before or after another program in the schedule. This option is only available for programs that use VOD sources.
  - **Absolute**: The program plays at a specific wall-clock time. MediaTailor makes a best effort to play the program at the clock time that you specify. MediaTailor starts playback of the program on a common segment boundary between the preceding program or slate. This option is only available for channels configured to use the linear **Playback mode**: The playback mode sets the channel's playback behavior. MediaTailor supports the following playback modes:.
  - **Program start time**: For absolute transition types, the wall-clock time when the program is scheduled to play. If you are adding this program to a running linear channel, you must enter a start time that's 15 minutes or later from the current time.
  - **Relative position**: Choose where to insert the program into the schedule, relative to another program. You can select **Before program** or **After program**. This setting does not apply if this is the first program in your channel's schedule.
    - If you choose **Select an existing program**, select the program name from a predefined list of the next 100 programs played by the channel from the **Use existing program** menu.
    - If you choose **Search for a program by name**, enter the name of an existing program in your channel.

If you'd like to add ad breaks to your program, continue to the next step. Ad breaks are only configurable for programs that use VOD sources. For live sources, ad breaks in DASH manifests and ad breaks in HLS manifests that use the `EXT-X-DATERANGE` tag are passed through automatically.

7. Select **Next** to go to **Add ad breaks**.
8. Select **Add ad break**. Under **Ad breaks**, configure the settings for the ad break:

   - **Slate source location name**: Choose **Select an existing source location** and choose the source location where your slate is stored that you created earlier in this task.
   - **VOD source name**: Choose **Select an existing VOD source** and choose the VOD source you're using for slate that you added earlier in this task. The duration of the slate determines the duration of the ad break.
   - **Offset in milliseconds**: This value determines the ad break start time in milliseconds, as an offset relative to the beginning of the program. Enter any value that's less than the duration of the VOD source, and that aligns with a segment boundary on all tracks within the program's VOD source (all audio, video and closed caption tracks), otherwise the ad break will be skipped. For example, if you enter **0**, this creates a pre-roll ad break that plays before the program begins.
   - **Avail number**: MediaTailor writes this value is written to `splice_insert.avail_num`, as defined in section 9.7.3.1. of the SCTE-35 specification, Digital Program Insertion Cueing Message. The default value is 0. Values have to be between 0 and 256, inclusive.
   - **Avail expected**: MediaTailor writes this value to `splice_insert.avails_expected`, as defined in section 9.7.3.1. of the SCTE-35 specification. The default value is 0. Values have to be between 0 and 256, inclusive.
   - **Splice event ID**: MediaTailor writes this value to `splice_insert.splice_event_id`, as defined in section 9.7.3.1. of the SCTE-35 specification. The default value is 1.
   - **Unique program ID**: MediaTailor writes this value to `splice_insert.unique_program_id`, as defined in section 9.7.3.1. of the SCTE-35 specification. The default value is 0. Values have to be between 0 and 256, inclusive.
9. For a Standard Linear Channel, select **Next** to go to **Set alternate media**.

   For more information on using MediaTailor to create alternate media, see Creating alternate media.

   For more advanced information on using MediaTailor to personalize your ad breaks, see Insert personalized ads and ad breaks in a channel stream.
10. Select **Next** to go to **Review and create**.
11. Select **Add program**.

For more advanced information on using MediaTailor to personalize your ad breaks, see Insert personalized ads and ad breaks in a channel stream.

12. **Important**

    For looping channels, if you modify the program list for a program that is scheduled within the next 10 minutes, the edit won't become apparent until the next loop.

    Under **Program details**, enter details about your program:

    - **Name**: This is the name of the program that you add to your channel.
    - **Source type**: Determines what type of source the program plays. This option is only available for Standard channels.
      - **VOD** - The program plays a VOD source, such as a pre-recorded TV episode.
      - **Live** - The program plays a live source, such as a live news broadcast.
    - **Source location name**: The source location to be associated with the program.

      If you choose **Select an existing source location**, select a source location name from the **Select a source location** drop-down menu. You can alternatively search for your source location by name. This is helpful if you have a large number of source locations.

      If you choose **Enter the source location name**, search for your source location by name.
    - **VOD source name**: The name of the VOD source to be associated with the program.

      If you choose **Select an existing VOD source**, select a VOD source name from the list of VOD sources that are associated with your account. You can alternatively search for your VOD source by name. This is helpful if you have a large number of VOD sources.

      If you choose **Search by name**, search for your VOD source by name.
    - **Live source name**: The name of the live source to be associated with the program. This option is only available if you selected **Live** as the source type.

      If you choose **Select an existing live source**, select a live source name from the list of live sources that are associated with your account. You can alternatively search for your live source by name. This is helpful if you have a large number of live sources.

      If you choose **Search by name**, search for your live source by name.

13. Under **Playback configuration**, define when a program plays in your channel's schedule:

- **Duration in milliseconds**: Defines the duration of the program in milliseconds. This option is only available for programs that use live sources.
- **Transition type**: Defines the transitions from program to program in the schedule.
  - **Relative** - The program plays either before or after another program in the schedule. This option is only available for programs that use VOD sources.
  - **Absolute** - The program plays at a specific wall clock time. MediaTailor makes a best effort to play the program at the clock time that you specify. We start playback of the program on a common segment boundary between the preceding program or slate. This option is only available for channels configured to use the linear playback mode.

    > **Note**
    >
    > Be aware of the following behavior for absolute transition types:
    >
    > - If the preceding program in the schedule has a duration that extends beyond the wall clock time, MediaTailor truncates the preceding program on the common segment boundary closest to the wall clock time.
    > - If there are gaps between programs in the schedule, MediaTailor plays filler slate. If the duration of the slate is less than the duration of the gap, MediaTailor loops the slate.

- **Program start time** - For absolute transition types, the wall clock time when the program is scheduled to play. If you are adding this program to a running linear channel, you must enter a start time that's 15 minutes or later from the current time.
- **Relative position**: Choose where to insert the program into schedule relative to another program. You can select **Before program** or **After program**. This setting does not apply if this is the first program in your channel's schedule.
- **Relative program**: The name of the program to be used to insert the new program before or after. This setting does not apply if this is the first program in your channel's schedule.

  If you choose **Select an existing program**, select the program name from a predefined list of the next 100 programs played by the channel in the **Use existing program** drop-down menu.

  If you choose **Search for a program by name**, enter name of an existing program in your channel.

If you'd like to add ad breaks to your program, continue to the next step. Ad breaks are only configurable for programs that use VOD sources. For live sources, ad breaks in DASH manifests and ad breaks in HLS manifests that use the `EXT-X-DATERANGE` tag are passed through automatically.

14. Select **Add ad break**. Under **Ad breaks**, configure the settings for the ad break:

    - **Slate source location name**: Choose **Select an existing source location** and choose the source location where your slate is stored that you created earlier in this tutorial.
    - **VOD source name**: Choose **Select an existing VOD source** and choose the VOD source you're using for slate that you added earlier in this tutorial. The duration of the slate determines the duration of the ad break.
    - For **Offset in milliseconds**: This value determines the ad break start time in milliseconds, as an offset relative to the beginning of the program. Enter any value that's less than the duration of the VOD source, and that aligns with a segment boundary on all tracks within the program's VOD source (all audio, video and closed caption tracks), otherwise the ad break will be skipped. For example, if you enter **0**, this creates a pre-roll ad break that plays before the program begins.

      > **Note**
      >
      > If MediaTailor detects ad markers, such as`DATERANGE` or `EXT-X-CUE-OUT` for HLS and `EventStream` for DASH, with durations of zero within your VOD source, you can select the offset of those ad markers from the drop-down menu to be used as the ad break's offset. In order for an ad opportunity to be detected, it must be present at the same offset across all package configurations within a VOD source, and its duration must be zero.

    - For **Avail number**, this is written to `splice_insert.avail_num`, as defined in section 9.7.3.1. of the SCTE-35 specification. The default value is `0`. Values have to be between `0` and 256, inclusive.

      For **Avail expected**, this is written to `splice_insert.avails_expected`, as defined in section 9.7.3.1. of the SCTE-35 specification. The default value is `0`. Values have to be between `0` and 256, inclusive.

For **Splice event ID**, this is written to `splice_insert.splice_event_id`, as defined in section 9.7.3.1. of the SCTE-35 specification. The default value is 1.

For **Unique program ID**, this is written to `splice_insert.unique_program_id`, as defined in section 9.7.3.1. of the SCTE-35 specification. The default value is `0`. Values have to be between `0` and 256, inclusive.

15. Choose **Add program**.

For more advanced information using MediaTailor to personalize your ad breaks, see Insert personalized ads and ad breaks in a channel stream.

> **Note**
>
> If your channel has at least one output with an `Enhanced Scte35` Ad markup type, you can submit ad-break metadata. MediaTailor writes the submitted key-value pairs to the `EXT-X-ASSET` tag for your ad break.

Defining audience cohorts and alternate content with Program Rules

With Program Rules, you can define audience cohorts for a channel and specify alternate media to play for those audiences. You can associate one or more alternate content sources with an audience for a program. After the program ends, the default audience content will play unless you specify further alternate media.

Program Rules are available on STANDARD tier channels with the LINEAR playback mode. MediaTailor channels support alternate media for all VOD sources and live sources.

For an example use, see Using program rules with AWS MediaTailor.

Defining audiences

Define audiences on a channel by typing audience one by one when configuring a MediaTailor channel. You can do this through either the MediaTailor console or the MediaTailor `CreateChannel` API. Each audience must be between 1 and 32 alphanumeric characters long. If the values provided for the audiences are invalid, then the request fails.

You can only define audiences on STANDARD tier channels with the LINEAR playback mode.

When you need to update the audiences, you can do this using either the MediaTailor console or the MediaTailor `UpdateChannel` API.

If you are using the `ProgramRules` feature, make sure that the `AudienceMedia` defined in `CreateProgram` or `UpdateProgram` request contain the existing audience defined in the channel.

Creating alternate media

The following task explains how to define alternate media using the MediaTailor console. For information about how to define alternate media using the MediaTailor API, see `CreateProgram` in the *AWS Elemental MediaTailor API Reference*.

To define alternate media on a new program:

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, select **Channel assembly** > **Channels**.
3. Select the channel name to which you want to add alternate media.
4. Create a program. For more information, see Creating a program within a channel schedule using the MediaTailor console.
5. Configure alternate media:

   - Select **Add** in the **Audiences** box to select the audience for which you are defining alternate media.
   - Select an audience defined on the channel from the **Audience** menu.
   - Select **Add alternate media** to begin defining alternate media for the program.
   - MediaTailor creates an **Alternate media 1** box. This is the first content that MediaTailor plays as alternate media on the program.
   - Within the **Alternate media 1** box:
     - Select a **Source Location**.
     - Select either a **VOD** or **Live** Source Type:

       For VOD

       - Select VOD for the **Source Type**.
       - (Optional) specify a **Clip Range**. With VOD Sources, including alternate-media VOD sources, you can specify a portion of a VOD source to play, clipping from the start and/or the end of the source. Specify The start and end offsets are in milliseconds.

- (Optional) a\Add Ad Breaks. This is done in the same way as when creating programs. For more information, see Creating a program within a channel schedule using the MediaTailor console.

For Live

- Select Live for the **Source Type**.
- Select a **Live source**.
- Enter a **Start time** in milliseconds of the wal-clock time that this live source should start. The live source will only play within the time frame of the default program it is being defined on. If the start time is prior to the start of the default program, it will not begin until the default program does. If the start time is after the default program ends, MediaTailor will not play the live source.
- Enter a **Duration** in milliseconds. The duration must be at least 10 minutes in length.
- Additional alternate media can be added to this program for the audience by selecting **Add alternate media** again. This will create another box labeled **Alternate media 2**. You can specify upp to 5 alternate-media sources per program, per audience.
- Once you are finished defining alternate media for all desired audiences, select **Next** and continue creating the program.

  For more information, see Creating a program within a channel schedule using the MediaTailor console.

> **Note**
>
> Alternate media only plays in the time frame of the program it is defined on. If all the alternate content overruns the default content, MediaTailor will truncate it. MediaTailor plays alternate media in the order in which it is defined. Live alternate-media start times will always take precedence and will truncate previously scheduled VOD sources or live sources. Any time that is not filled with alternate media for an audience will be filled withe the channel-defined filler slate

- To define audience media for other audiences, select **Add** once again next to **Audiences**. Select the newly created audience, set the audience id and add alternate media as described above. Up to 5 audiences can have alternate media on any one program.

Generating audience-specific manifests

To retrieve a manifest for a particular audience, use the `aws.mediatailor.channel.audienceId` query parameter. This query parameter may be dynamically appended by your CDN or added through a call to your content or customer-management system. You must maintain the association of a given playback session to an `audienceId` outside of MediaTailor. This will retrieve an audience-specific manifest with any alternate media defined for that audience in place of default content. It is important that once a manifest is requested for a particular audience that the player always requests the manifest with that same audience ID or there could be playback errors.

If a request is made for an audience that does not exist on the channel, MediaTailor returns a 404 error.

Example Getting a manifest for an audience

```
https:// prefix>.channel-assembly.mediatailor.us-
west-2.amazonaws.com/v1/channel/ExampleChannel/index_dash.mpd?
aws.mediatailor.channel.audienceId=Seattle
```

Insert personalized ads and ad breaks in a channel stream

With MediaTailor, you can monetize channel assembly linear streams by inserting ad breaks in your programs without conditioning the content with SCTE-35 markers. You can use channel assembly with the MediaTailor ad insertion service, or with any server-side ad insertion (SSAI).

The following topics show how to insert personalized ads and ad breaks in your channel's linear stream.

Topics

- Setting up ad insertion with MediaTailor
- SCTE-35 messages for ad breaks

Setting up ad insertion with MediaTailor

To insert personalized ads into your channel's stream, your channel's endpoint URL is the content source for AWS Elemental MediaTailor. This guide shows how to set up MediaTailor for ad insertion.

Prerequisites

Before you begin, make sure that you meet the following requirements:

- Prepare your HLS and DASH streams for MediaTailor ad insertion.
  - If you haven't prepared content streams already, see Step 2: Prepare a stream in the *Getting started with MediaTailor ad insertion* topic.
- Have an ad decision server (ADS).
- Configure **Ad break** settings in the program. For more information, see the Configuring ad breaks for your program procedure.

As a best practice, consider using a content delivery network (CDN) in between channel assembly and MediaTailor ad insertion. The MediaTailor ad insertion service can generate additional origin requests. Therefore, it's a best practice to configure your CDN to proxy the manifests from channel assembly, then use the CDN prefixed URLs at the content source URL.

Configure MediaTailor for ad insertion

The following shows how to configure MediaTailor console settings so that you can insert personalized ads into your channel's stream.

To configure MediaTailor for ad insertion

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Configurations**.
3. Under **Required settings**, enter the basic required information about your configuration:
   - **Name**: The name of your configuration.
   - **Content source**: Enter the playback URL from your channel's output, minus the file name and extension. For advanced information about MediaTailor configuration, see Required settings.
   - **Ad decision server**: Enter the URL for your ADS.
4. You can optionally configure the **Configuration aliases**, **Personalization details**, and **Advanced settings**. For information about those settings, see Optional configuration settings.
5. On the navigation bar, choose **Create configuration**.

Now that you've set up MediaTailor for ad insertion, you can also set up ad breaks. For detailed instructions, see Getting started with MediaTailor ad insertion.

SCTE-35 messages for ad breaks

With MediaTailor, you can create a content channel based off of source location and VOD source resources. You can then set up one or more ad breaks for each of the programs on a channel's schedule. You use messages based on the SCTE-35 specification to condition the content for ad breaks. For example, you can use SCTE-35 messages to provide metadata about the ad breaks. For more information about the SCTE-35 specification, see Digital Program Insertion Cueing Message.

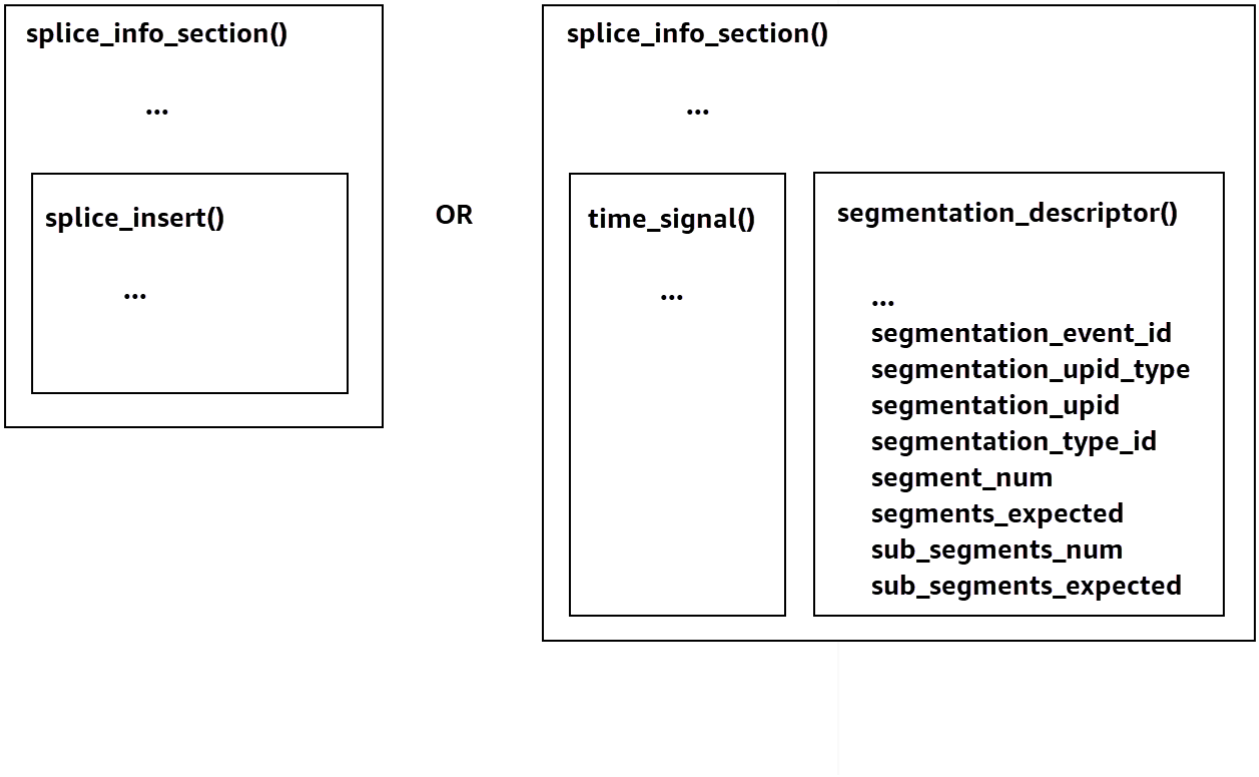
You set up the ad breaks in one of two ways:

- Attaching a `time_signal` SCTE-35 message with a `segmentation_descriptor` message. This `segmentation_descriptor` message contains more advanced metadata fields, like content identifiers, which convey more information about the ad break. MediaTailor writes the ad metadata to the output manifest as part of the `EXT-X-DATERANGE` (HLS) or `EventStream` (DASH) ad marker's SCTE-35 data.
- Attaching a `splice_insert` SCTE-35 message that provides basic metadata about the ad break.
- HLS:
  - When the Ad markup type is `Daterange`, MediaTailor specifies ad breaks as `EXT-X-DATERANGE` tags in the manifest.
  - When the Ad markup type is `Scte35 Enhanced`, MediaTailor specifies ad breaks using the following tags:
    - MediaTailor places an `EXT-X-CUE-OUT` on the first segment of the ad slate, indicating a cut from content to the ad break. It contains the expected duration of ad break, such as `EXT-X-CUE-OUT:Duration=30`.
    - `>EXT-X-ASSET`: This tag appears on the same segment as `EXT-X-CUE-OUT` and contains the ad-break metadata supplied in the AdBreak when the program is created or updated. It always contains `CAID`.
    - `EXT-OATCLS-SCTE35`: This tag appears on the same segment as `EXT-X-CUE-OUT` and contains base64-encoded bytes of the SCTE-35 message.
    - `EXT-X-CUE-OUT-CONT`: This tag appears on each subsequent segment within the ad slate, and contains duration and elapsed-time information. It also contains the base64-encoded SCTE-35 message, and the `CAID`.

- `EXT-X-CUE-IN`: This tag appears on the first segment of content after the ad break is over, and indicates a cut from an ad break back to content.

The following illustration shows the two ways of setting up ad breaks in a channel using SCTE-35 messages:

- Use a `splice_insert()` message to set up ad breaks with basic metadata.
- Use a `time_signal()` message together with a `segmentation_descriptor()` message to set up ad breaks with more detailed metadata.



For information about using `time_signal`, see section 9.7.4 of the 2022 SCTE-35 specification, Digital Program Insertion Cueing Message.

The ad break information appears in the output `splice_info_section` SCTE-35 data. With MediaTailor, you can pair a single `segmentation_descriptor` message together with a single `time_signal` message.

Note

If you send a `segmentation_descriptor` message, you must send it as part of the `time_signal` message type. The `time_signal` message contains only the `splice_time` field that MediaTailor constructs using a given timestamp.

The following table describes the fields that MediaTailor requires for each `segmentation_descriptor` message. For more information, see section 10.3.3.1 of the 2022 SCTE-35 specification, which you can purchased at the ANSI Webstore website.

Required fields for a `segmentation_descriptor` message

Field	Type	Default value	Description
`segmentation_event_id`	integer	1	This is written to `segmentation_descriptor.segmentation_event_id`.
`segmentation_upid_type`	integer	14 (0x0E)	This is written to `segmentation_descriptor.segmentation_upid_type`. The value must be between 0 and 256, inclusive.
`segmentation_upid`	string	"" (empty string)	This is written to `segmentation_descriptor.segmentation_upid`. The value must be a

Field	Type	Default value	Description
			hexadecimal string, containing characters `0-9` and `A-F`.
`segmentation_type_id`	integer	48 (0x30)	This is written to `segmentation_descriptor.segmentation_type_id` . The value must be between 0 and 256, inclusive.
`segment_num`	integer	0	This is written to `segmentation_descriptor.segment_num` . The value must be between 0 and 256, inclusive.
`segments_expected`	integer	0	This is written to `segmentation_descriptor.segments_expected` . The value must be between 0 and 256, inclusive.

Field	Type	Default value	Description
sub_segment_num	integer	null	This is written to segmentation_descriptor.sub_segment_num. The value must be between 0 and 256, inclusive.
sub_segments_expected	integer	null	This is written to segmentation_descriptor.sub_segments_expected. The value must be between 0 and 256, inclusive.

The following table shows the values that MediaTailor automatically sets for some of the segmentation_descriptor message's fields.

Values set by MediaTailor for a segmentation_descriptor message's fields

Field	Type	Value
segmentation_event_cancel_indicator	Boolean	True
program_segmentation_flag	Boolean	True
delivery_not_restricted_flag	Boolean	True

MediaTailor always sets the `segmentation_duration_flag` to `True`. MediaTailor populates the `segmentation_duration` field with the duration, in ticks, of the state content.

> **Note**
>
> When MediaTailor sends the `time_signal` messages, it sets the `splice_command_type` field in the `splice_info_section` message to 6 (0x06).

In HLS output, for an `AdBreak` with a `time_signal` message, the output `EXT-X-DATERANGE` tag includes a `SCTE-35` field that is set to the serialized version of the `splice_info_section` message. For example, the following `EXT-X-DATERANGE` tag shows the serialized version of the `splice_info_section` message:

```
#EXT-X-DATERANGE:ID=\"1\",START-DATE=\"2020-09-25T02:13:20Z\",DURATION=300.0,SCTE35-
OUT=0xFC002C0000000000000FFF0050680000000000016021443554549000000100E000019BFCC00E00300000000
```

In DASH output, for an `AdBreak` with a `time_signal` message, the output `EventStream` element includes a `scte35:SpliceInfoSection` element with `scte35:TimeSignal` and `scte35:SegmentationDescriptor` elements as its children. The `scte35:TimeSignal` element has a child `scte35:SpliceTime` element, and the `scte35:SegmentationDescriptor` element has a child `scte35:SegmentationUpid` element. For example, the following DASH output shows the `EventStream` element's child element structure:

```
<EventStream schemeIdUri="urn:scte:scte35:2013:xml" timescale="90000">
    <Event duration="27000000">
        <scte35:SpliceInfoSection protocolVersion="0" ptsAdjustment="0" tier="4095">
            <scte35:TimeSignal>
                <scte35:SpliceTime ptsTime="0" />
            </scte35:TimeSignal>
            <scte35:SegmentationDescriptor segmentNum="0"
 segmentationDuration="27000000" segmentationEventCancelIndicator="false"
 segmentationEventId="1" segmentationTypeId="48" segmentsExpected="0">
                <scte35:SegmentationUpid segmentationUpidFormat="hexBinary"
 segmentationUpidType="14">012345</scte35:SegmentationUpid>
            </scte35:SegmentationDescriptor>
        </scte35:SpliceInfoSection>
    </Event>
</EventStream>
```

You learned about using SCTE-35 messages to set up ad breaks in channel assembly, the structure and required fields for those messages, and sample HLS and DASH output that includes the SCTE-35 messages.

Enable time-shifted viewing

Time-shifted viewing means that viewers can start watching a live stream at a time earlier than the time of request, permitting them to join from the beginning a program that's already in progress or to watch a program that's already completed. MediaTailor channels support time-shifted viewing for content that's up to 6 hours old. You can enable time-shifted viewing for some or all of this content by defining the **maximum time delay** on the channel. Clients can shift the manifest window back in time, by up to the configured maximum time delay, by including valid time-shifting parameters on requests.

Time-shifted viewing is available on STANDARD tier channels with the LINEAR playback mode. MediaTailor channels support time shifting for all VOD sources, and for live sources that utilize MediaPackage V1 or MediaPackage V2 origins configured with sufficient startover windows.

> **Important**
>
> MediaPackage V1 and MediaPackage V2 channels used as live origins for MediaTailor channels must have sufficient startover windows. We recommend a startover window that is at least 10 minutes longer than the sum of the maximum time delay and the manifest-window duration on your MediaTailor channel.
> For information on how to configure a startover window on a MediaPackage V1 channel, refer to Time-shifted viewing reference in AWS Elemental MediaPackage in the *MediaPackage V1 user guide*. For information on how to configure a startover window on a MediaPackage V2 channel, refer to Time-shifted viewing reference in AWS Elemental MediaPackage in the *MediaPackage V2 user guide*.

To enable time-shifted viewing

1. Enable time-shifted viewing by typing a value for **Maximum time delay** when configuring a MediaTailor channel. You can do this through either the MediaTailor console or the MediaTailor API. The minimum allowed maximum time delay is 0 seconds, and the maximum allowed maximum time delay is 21600 seconds (6 hours).

When MediaTailor receives requests for this channel with time shifting parameters that conform to the configured maximum time delay, MediaTailor generates a sliding window manifest starting at the specified time. If the values provided for the time-shifting parameters require a delay that exceeds the maximum time delay, then the requests fail. If the request has no time-shifting parameters, the service generates a manifest with no delay.

2. Make sure that content requests contain time-shifting parameters as needed. A request can have up to one time-shifting parameter. For information on specific time shifting parameters, see Time-shifting parameters for manifest requests.

Topics

- Time-shifting parameters for manifest requests
- Using time-shifted viewing with CDNs

Time-shifting parameters for manifest requests

This section lists the parameters for time-shifting manifest requests.

To use this functionality, follow the steps in Enable time-shifted viewing.

Time delay

You can specify a duration in seconds for MediaTailor to delay when content is available to players. The minimum is 0 seconds, and the maximum is the maximum time delay that you have configured for the channel.

Use the `aws.mediatailor.channel.timeDelay` parameter to redefine the live point and make content available later than when it appears in your channel's schedule. With a 60-second time delay, content that appears in MediaTailor's schedule at 12:20 is not available until 12:21. Likewise, if you are serving content across time zones, you can set a time delay equal to the difference to make content available at, for example, 8:00 local time.

To supply a time delay to a manifest request, include `aws.mediatailor.channel.timeDelay` as a query parameter.

Example time delay

```
https://<some prefix>.channel-assembly.mediatailor.us-west-2.amazonaws.com/v1/channel/
ExampleChannel/index_dash.mpd?aws.mediatailor.channel.timeDelay=901
```

Start time

You can specify a timestamp from which to start playback using the `aws.mediatailor.channel.startTime` parameter. The start time must be specified in one of the following formats:

- ISO 8601 dates, such as `2017-08-18T21:18:54+00:00`

  Any + symbols in ISO 8601 dates must be URL-encoded as %2B, such as `2017-08-18T21:18:54%2B00:00`
- POSIX (or Epoch) time, such as `1503091134`

When provided with a start time, MediaTailor responds with a sliding-window manifest, as if the player had requested the initial manifest at the specified start time. For example, a viewer who starts watching a channel at `2023-10-25T14:00:00` and provides an `aws.mediatailor.channel.startTime` of `2023-10-25T12:00:00` sees the same content as a viewer who starts watching the same channel at `2023-10-25T12:00:00` with no start time specified. On channels with the LINEAR playback mode, the last segment in the manifest window is the segment that overlaps with the time 10 seconds before the time that the request is made. In addition, players maintain a buffer between the playback point and the end of the manifest window. Therefore, playback does not start exactly with the content scheduled for the specified start time.

The delay that results from the specified start time must be at least 0 and must be no greater than the maximum time delay that you configured for the channel.

Example start time

```
https://<some prefix>.channel-assembly.mediatailor.us-west-2.amazonaws.com/v1/channel/
ExampleChannel/
index_dash.mpd?aws.mediatailor.channel.startTime=2017-12-19T13:00:28-08:00
```

Start program

You can specify a program from which playback should start using the `aws.mediatailor.channel.startProgram` parameter. Acceptable values are the names of

programs whose start times fall within the the maximum time delay that you configured for the channel from the end of the manifest window.

When provided with a start program, MediaTailor indicates to players the exact point at which playback should start.

- For HLS outputs, MediaTailor chooses a delay such that the first segment of the start program is 29 seconds from the end of the manifest window, and uses an `EXT-X-START` tag in the primary manifest to indicate that the player should start playback with the segment that is 29 seconds from the end of the manifest window.
- For DASH outputs, MediaTailor chooses a delay based on the suggested presentation delay that you have configured for your output. If your output has a non-zero suggested presentation delay, MediaTailor chooses a delay where the duration between the start of the first segment of the start program and the wall clock time that MediaTailor receives the request is equal to the output's suggested presentation delay. Otherwise, MediaTailor chooses a delay where the first segment of the start program is 29 seconds from the end of the manifest window. For the best results, we recommend that you configure your output to have a suggested presentation delay that is at least three times the maximum segment duration on your start program, plus 10 seconds.

Players do not necessarily obey MediaTailor's suggestions, and the point at which playback starts may vary slightly depending on what player you use and how you configure it. We recommend that you test your channel with a start program parameter in your player and, if necessary, make adjustments to your player's configuration so that it starts playback at the first segment of the start program.

Example start program

```
https://<ome prefix>.channel-assembly.mediatailor.us-west-2.amazonaws.com/v1/channel/
ExampleChannel/index_dash.mpd?aws.mediatailor.channel.startProgram=SuperBowlLVII
```

Using time-shifted viewing with CDNs

To achieve a sliding window when provided with a start time or start program, MediaTailor translates the start time or start program value into an appropriate time delay. The value of that time delay depends on the time at which the player or CDN requests the manifest. Because of this, when using a CDN with MediaTailor's start time or start program parameters, you must configure the appropriate caching behavior on your CDN.

To use this functionality, follow the steps in Enable time-shifted viewing.

HLS example

Suppose that you request an HLS primary manifest with a start time using a URL like the one below:

```
https://<some prefix>.channel-assembly.mediatailor.us-west-2.amazonaws.com/v1/channel/
ExampleChannel/
index_hls.m3u8?aws.mediatailor.channel.startTime=2017-12-19T13:00:28-08:00
```

MediaTailor responds with a manifest that includes time-delay parameters on the child manifest URLs. For example, if you request the manifest at time `2017-12-19T13:20:28-08:00`, which is 1200 seconds after the requested start time, then MediaTailor responds with a primary manifest like the one below:

```
#EXTM3U
#EXT-X-VERSION:6
#EXT-X-STREAM-INF:CODECS="avc1.4D401F,mp4a.40.2",AVERAGE-
BANDWIDTH=1426714,RESOLUTION=852x480,FRAME-RATE=30.0,BANDWIDTH=1493368
index_hls/1.m3u8?aws.mediatailor.channel.timeDelay=1200
#EXT-X-STREAM-INF:CODECS="avc1.4D401E,mp4a.40.2",AVERAGE-
BANDWIDTH=986714,RESOLUTION=640x360,FRAME-RATE=30.0,BANDWIDTH=1024034
index_hls/2.m3u8?aws.mediatailor.channel.timeDelay=1200
#EXT-X-STREAM-INF:CODECS="avc1.4D400D,mp4a.40.2",AVERAGE-
BANDWIDTH=476305,RESOLUTION=320x240,FRAME-RATE=30.0,BANDWIDTH=498374
index_hls/3.m3u8?aws.mediatailor.channel.timeDelay=1200
```

DASH example

Suppose that you request a DASH manifest with a start time using a URL like the one below:

```
https://<some prefix>.channel-assembly.mediatailor.us-west-2.amazonaws.com/v1/channel/
ExampleChannel/
index_dash.mpd?aws.mediatailor.channel.startTime=2017-12-19T13:00:28-08:00
```

MediaTailor responds with a redirect to the same manifest, but with a time delay instead of a start time. For example, if you request the manifest at time `2017-12-19T13:20:28-08:00`, which is 1200 seconds after the requested start time, then MediaTailor responds with HTTP status `302 Found` and a `Location` header with value `./index_dash.mpd?aws.mediatailor.channel.timeDelay=1200`.

CDN configuration requirements

When using time-shifting query parameters with a CDN, we recommend that you configure your CDN as follows:

- If you use any time shifting query parameters, include those parameters in your CDN's cache key. Additionally, include the time-delay query parameter in your CDN's cache key if you use any time-shifting parameters.
- If you use one of the start-time or start-program query parameters, then the following apply:
  - For HLS, configure your CDN to cache primary manifests for no longer than a typical segment duration on your channel.
  - For DASH, configure your CDN to cache redirects with HTTP status 302 for no longer than a typical segment duration on your channel, and to forward such redirects to the player.

For information on how to configure caching on Amazon CloudFront, refer to Managing how long content stays in the cache (expiration) in the *CloudFront Developer guide*. For information on how Amazon CloudFront handles redirects, refer to How CloudFront processes HTTP 3xx status codes from your origin. in the *CloudFront Developer guide*.

Troubleshooting playback errors returned by MediaTailor

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

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

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

AWS Elemental MediaTailor returns two classes of errors:

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

Topics

- Client playback errors returned by AWS Elemental MediaTailor
- Server playback errors returned by AWS Elemental MediaTailor
- Playback error examples

Client playback errors returned by AWS Elemental MediaTailor

General guidance:

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

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

Co	Exception name	Meaning	What to do
40	`BadReques tException`	MediaTailor is unable to service the request due to one or more errors in formatting or content. A parameter might be improperl y formatted, or the request	Check that your request is properly formatted and contains accurate information. Make sure that the playback endpoint setting on the player matches the `ManifestE`

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

If you need further assistance, contact AWS Support.

Server playback errors returned by AWS Elemental MediaTailor

General guidance:

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

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

Co	Exception name	Meaning	What to do
50	`InternalServiceError`	Unhandled exception.	Retry the request. If the problem persists, check the reported health of MediaTailor for your AWS Region at https://status.aws.amazon.com/.
50	`BadGatewayException`	Either the origin server address or the ad decision server (ADS) address is invalid. Examples of invalid addresses are a private IP address and `localhost` .	Make sure that your configuration has the correct settings for your ADS and origin server, and then retry the request.
50	`UnsupportedManifestException`	Either the origin manifest has changed so that MediaTailor can't personalize it or MediaTail or doesn't support the origin's manifest format.	This might affect only the individual session. Reinitialize the session. You can usually accomplish this by refreshing the page in the viewer. If the problem persists, verify that MediaTailor supports the origin's manifest format. For

Co	Exception name	Meaning	What to do
			information, see Integrating a content source.
50	LoadShed	MediaTailor experienced a resource constraint while servicing your request.	Retry the request. If the problem persists, check the reported health of MediaTailor for your AWS Region at https://status.aws.amazon.com/.
50	ThrottlingException	Your transactions per second have reached your quota, and MediaTailor is throttling your use.	Retry the request. You can also check the reported health of MediaTailor for your AWS Region at https://status.aws.amazon.com/. You might want to increase the quota on your transactions per second. For more information, see the section called "Quotas on ad insertion".
50	GatewayTimeoutException	A timeout occurred while MediaTailor was contacting the origin server.	Retry the request. If the problem persists, check the health of the origin server and make sure the origin server is responding within the content origin server timeout that is listed at the section called "Quotas on ad insertion".

If you need further assistance, contact AWS Support.

Playback error examples

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

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

```
~[]> curl -vvv https://111122223333444455556666123456789012.mediatailor.us-
west-2.amazonaws.com/v1/master/123456789012/Multiperiod_DASH_Demo/index.mpd
*   Trying 54.186.133.224...
* Connected to 111122223333444455556666123456789012.mediatailor.us-west-2.amazonaws.com
 (11.222.333.444) port 555 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: mediatailor.us-west-2.amazonaws.com
* Server certificate: Amazon
* Server certificate: Amazon Root CA 1
* Server certificate: Starfield Services Root Certificate Authority - G2
> GET /v1/master/123456789012/Multiperiod_DASH_Demo/index.mpd HTTP/1.1
> Host: 111122223333444455556666123456789012.mediatailor.us-west-2.amazonaws.com
> User-Agent: curl/7.43.0
> Accept: */*
>
< HTTP/1.1 504 Gateway Timeout
< Date: Thu, 29 Nov 2018 18:43:14 GMT
< Content-Type: application/json
< Content-Length: 338
< Connection: keep-alive
< x-amzn-RequestId: 123456789012-123456789012
< x-amzn-ErrorType: GatewayTimeoutException:http://internal.amazon.com/coral/
com.amazon.elemental.midas.mms.coral/
<
* Connection #0 to host 111122223333444455556666123456789012.mediatailor.us-
west-2.amazonaws.com left intact
{"message":"failed to generate manifest: Unable to obtain template playlist.
 origin URL:[https://777788889999.mediapackage.us-west-2.amazonaws.com/out/
v1/4444555566661111222233333/index.mpd], asset path: [index.mpd], sessionId:
[12345678901212345678901 2] customerId:[123456789012]"}%
```

Security in AWS Elemental MediaTailor

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

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

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

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

Topics

- Data protection in AWS Elemental MediaTailor
- Identity and Access Management for AWS Elemental MediaTailor
- Compliance validation for AWS Elemental MediaTailor
- Resilience in AWS Elemental MediaTailor
- Infrastructure security in MediaTailor
- Cross-service confused deputy prevention
- Logging and monitoring in MediaTailor

Data protection in AWS Elemental MediaTailor

The AWS shared responsibility model applies to data protection in AWS Elemental MediaTailor. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Data encryption

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

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

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

Identity and Access Management for AWS Elemental MediaTailor

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use MediaTailor resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How AWS Elemental MediaTailor works with IAM
- Identity-based policy examples for AWS Elemental MediaTailor
- Resource-based policy examples for AWS Elemental MediaTailor
- AWS managed policies for AWS Elemental MediaTailor
- Using service-linked roles for MediaTailor
- Troubleshooting AWS Elemental MediaTailor identity and access

Audience

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

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

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

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to MediaTailor. To view example MediaTailor identity-based policies that you can use in IAM, see Identity-based policy examples for AWS Elemental MediaTailor.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

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

AWS account root user

When you create an AWS account, you begin with one 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 don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

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

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

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

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

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.

- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Use an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

Managing access using policies

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

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

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

Identity-based policies

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

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

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that

support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

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

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

- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

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

How AWS Elemental MediaTailor works with IAM

Before you use IAM to manage access to MediaTailor, learn what IAM features are available to use with MediaTailor.

IAM features you can use with AWS Elemental MediaTailor

IAM feature	MediaTailor support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes
Policy resources	No
Policy condition keys (service-specific)	Yes

IAM feature	MediaTailor support
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level view of how MediaTailor and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for MediaTailor

Supports identity-based policies: Yes

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

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for MediaTailor

To view examples of MediaTailor identity-based policies, see Identity-based policy examples for AWS Elemental MediaTailor.

Resource-based policies within MediaTailor

Supports resource-based policies: Yes

The MediaTailor service supports only one type of resource-based policy. It's called a *channel policy* because it's attached to a channel. This policy defines which principals can perform actions on the channel.

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

To learn how to attach a resource-based policy to a channel, see **Create a channel using the MediaTailor console**.

Resource-based policy examples within MediaTailor

To view examples of MediaTailor resource-based policies, see Resource-based policy examples for AWS Elemental MediaTailor.

Policy actions for MediaTailor

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API

operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of MediaTailor actions, see Actions defined by AWS Elemental MediaTailor in the *Service Authorization Reference*.

Policy actions in MediaTailor use the following prefix before the action:

```
mediatailor
```

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

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

To view examples of MediaTailor identity-based policies, see Identity-based policy examples for AWS Elemental MediaTailor.

Policy resources for MediaTailor

Supports policy resources: No

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

Policy condition keys for MediaTailor

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

For a list of MediaTailor condition keys, see Condition keys for AWS Elemental MediaTailor in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS Elemental MediaTailor.

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

ACLs in MediaTailor

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with MediaTailor

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

For MediaTailor, use the value **Partial**.

Using temporary credentials with MediaTailor

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switch from a user to an IAM role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for MediaTailor

Supports forward access sessions (FAS): Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a

different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for MediaTailor

Supports service roles: No

AWS Elemental MediaTailor doesn't support service roles.

Service-linked roles for MediaTailor

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing MediaTailor service-linked roles, see Using service-linked roles for MediaTailor.

Identity-based policy examples for AWS Elemental MediaTailor

By default, users and roles don't have permission to create or modify MediaTailor resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by MediaTailor, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Elemental MediaTailor in the *Service Authorization Reference*.

Topics

- Policy best practices
- Using the MediaTailor console
- Allow users to view their own permissions

Policy best practices

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

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the MediaTailor console

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

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

To ensure that users and roles can still use the MediaTailor console, also attach the MediaTailor*ConsoleAccess* or *ReadOnly* AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

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

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

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

Resource-based policy examples for AWS Elemental MediaTailor

To learn how to attach a resource-based policy to a channel, see **Create a channel using the MediaTailor console**.

Topics

- Anonymous access
- Cross-account access

Anonymous access

Consider the following `Allow` policy. With this policy in effect, MediaTailor allows anonymous access to the `mediatailor:GetManifest` action on the channel resource in the policy. This occurs where *`region`* is the AWS Region, *`accountID`* is your AWS account ID, and *`channelName`* is the name of the channel resource.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAnonymous",
      "Effect": "Allow",
```

```
      "Principal": "*",
      "Action": "mediatailor:GetManifest",
      "Resource": "arn:aws:mediatailor:region:accountID:channel/channelName"
    }
  ]
}
```

Cross-account access

Consider the following `Allow` policy. With this policy in effect, MediaTailor allows the `mediatailor:GetManifest` action on the channel resource in the policy, across accounts. This occurs where *`region`* is the AWS Region, *`accountID`* is your AWS account ID, and *`channelName`* is the name of the channel resource.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCrossAccountAccess",
      "Effect": "Allow",
      "Principal": {"AWS":  "arn:aws:iam::111111111111:root"},
      "Action": "mediatailor:GetManifest",
      "Resource": "arn:aws:mediatailor:region:accountID:channel/channelName"
    }
  ]
}
```

AWS managed policies for AWS Elemental MediaTailor

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles)

where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

AWS managed policy: AWSElementalMediaTailorFullAccess

You can attach the `AWSElementalMediaTailorFullAccess` policy to your IAM identities. It's useful for users who need to create and manage playback configurations and channel assembly resources, such as programs and channels. This policy grants permissions that allow full access to AWS Elemental MediaTailor. These users can create, update, and delete MediaTailor resources.

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

AWS managed policy: AWSElementalMediaTailorReadOnly

You can attach the `AWSElementalMediaTailorReadOnly` policy to your IAM identities. It's useful for users who need to view playback configurations and channel assembly resources, such as programs and channels. This policy grants permissions that allow read-only access to AWS Elemental MediaTailor. These users can't create, update, or delete MediaTailor resources.

```
{
 "Version": "2012-10-17",
 "Statement": {
  "Effect": "Allow",
  "Action": [
   "mediatailor:List*",
   "mediatailor:Describe*",
   "mediatailor:Get*"
```

```
  ],
  "Resource": "*"
 }
}
```

MediaTailor updates to AWS managed policies

View details about updates to AWS managed policies for MediaTailor since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the MediaTailor Document history for AWS Elemental MediaTailor.

Change	Description	Date
MediaTailor added new managed policies	MediaTailor added the following managed policies: • AWSElementalMediaTailorReadOnly • AWSElementalMediaTailorFullAccess	November 24, 2021
MediaTailor started tracking changes	MediaTailor started tracking changes for its AWS managed policies.	November 24, 2021

Using service-linked roles for MediaTailor

AWS Elemental MediaTailor uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to MediaTailor. Service-linked roles are predefined by MediaTailor and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up MediaTailor easier because you don't have to manually add the necessary permissions. MediaTailor defines the permissions of its service-linked roles, and unless defined otherwise, only MediaTailor can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your MediaTailor resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Supported Regions for MediaTailor service-linked roles

MediaTailor supports using service-linked roles in all of the regions where the service is available. For more information, see AWS Regions and Endpoints.

Topics

- Service-linked role permissions for MediaTailor
- Creating a service-linked role for MediaTailor
- Editing a service-linked role for MediaTailor
- Deleting a service-linked role for MediaTailor

Service-linked role permissions for MediaTailor

MediaTailor uses the service-linked role named **AWSServiceRoleForMediaTailor** – MediaTailor uses this service-linked role to invoke CloudWatch to create and manage log groups, log streams, and log events. This service-linked role is attached to the following managed policy: `AWSMediaTailorServiceRolePolicy`.

The AWSServiceRoleForMediaTailor service-linked role trusts the following services to assume the role:

- `mediatailor.amazonaws.com`

The role permissions policy allows MediaTailor to complete the following actions on the specified resources:

- Action: `logs:PutLogEvents` on `arn:aws:logs:*:*:log-group:/aws/MediaTailor/*:log-stream:*`
- Action: `logs:CreateLogStream, logs:CreateLogGroup, logs:DescribeLogGroups, logs:DescribeLogStreams` on `arn:aws:logs:*:*:log-group:/aws/MediaTailor/*`

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-Linked Role Permissions in the *IAM User Guide*.

Creating a service-linked role for MediaTailor

You don't need to manually create a service-linked role. When you enable session logging in the AWS Management Console, the AWS Command Line Interface (AWS CLI), or the AWS API, MediaTailor creates the service-linked role for you.

> **Important**
>
> This service-linked role can appear in your account if you completed an action in another service that uses the features supported by this role. Also, if you were using the MediaTailor service before September 15, 2021, when it began supporting service-linked roles, then MediaTailor created the AWSServiceRoleForMediaTailor role in your account. To learn more, see A New Role Appeared in My IAM Account.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you enable session logging, MediaTailor creates the service-linked role for you again.

You can also use the IAM console to create a service-linked role with the **MediaTailor** use case. In the AWS CLI or the AWS API, create a service-linked role with the `mediatailor.amazonaws.com` service name. For more information, see Creating a Service-Linked Role in the *IAM User Guide*. If you delete this service-linked role, you can use this same process to create the role again.

Editing a service-linked role for MediaTailor

MediaTailor does not allow you to edit the AWSServiceRoleForMediaTailor service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a Service-Linked Role in the *IAM User Guide*.

Deleting a service-linked role for MediaTailor

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don’t have an unused entity that is not actively monitored

or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If the MediaTailor service is using the role when you try to clean up the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To clean up MediaTailor resources used by the AWSServiceRoleForMediaTailor

- Before you can delete the service-linked role created by MediaTailor for the log configuration, you must first *deactivate* all of the log configurations in your account. To deactivate a log configuration, set the **percent enabled** value to **0**. This turns off all session logging the corresponding playback configuration. For more information, see Deactivating a log configuration.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS Command Line Interface (AWS CLI), or the AWS API to delete the AWSServiceRoleForMediaTailor service-linked role. For more information, see Deleting a Service-Linked Role in the *IAM User Guide*.

Troubleshooting AWS Elemental MediaTailor identity and access

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

Topics

- I am not authorized to perform an action in MediaTailor
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my MediaTailor resources

I am not authorized to perform an action in MediaTailor

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `mediatailor:`*`GetWidget`* permissions.

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

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `mediatailor:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to MediaTailor.

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

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

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

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my MediaTailor resources

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

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

To learn more, consult the following:

- To learn whether MediaTailor supports these features, see How AWS Elemental MediaTailor works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Compliance validation for AWS Elemental MediaTailor

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

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

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security Compliance & Governance – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- HIPAA Eligible Services Reference – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map

the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).

- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in AWS Elemental MediaTailor

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

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

In addition to the AWS global infrastructure, MediaTailor offers several features to help support your data resiliency and backup needs.

Infrastructure security in MediaTailor

As a managed service, AWS Elemental MediaTailor is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud

Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access MediaTailor through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

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

You can call these API operations from any network location, but MediaTailor does support resource-based access policies, which can include restrictions based on the source IP address. You can also use MediaTailor policies to control access from specific Amazon Virtual Private Cloud (Amazon VPC) endpoints or specific VPCs. Effectively, this isolates network access to a given MediaTailor resource from only the specific VPC within the AWS network.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the aws:SourceArn and aws:SourceAccount global condition context keys in resource policies to limit the permissions that AWS Elemental MediaTailor gives another service to the resource. If you use both global condition context keys, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement.

The value of `aws:SourceArn` must be the playback configuration that publishes CloudWatch logs for in your Region and account. However, this only applies if you use the `MediaTailorLogger` role that lets MediaTailor publish Amazon CloudWatch logs to your account. This doesn't apply if you use a service-linked role to let MediaTailor publish the CloudWatch logs.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full ARN of the resource. If you don't know the full ARN of the resource or if you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN. For example, `arn:aws:servicename::123456789012:*`.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in to prevent the confused deputy problem.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "mediatailor.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:mediatailor:region:account_ID:playbackConfiguration/
*"
      },
      "StringEquals": {
        "aws:SourceAccount": "account_ID"
      }
    }
  }
}
```

Logging and monitoring in MediaTailor

This section provides an overview of the options for logging and monitoring in AWS Elemental MediaTailor for security purposes. For more information about logging and monitoring in MediaTailor see Monitoring and tagging AWS Elemental MediaTailor resources.

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

Amazon CloudWatch Alarms

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

AWS CloudTrail logs

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

AWS Trusted Advisor

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

For more information, see AWS Trusted Advisor.

Monitoring and tagging AWS Elemental MediaTailor resources

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

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

Topics

- Viewing AWS Elemental MediaTailor logs
- Monitoring AWS Elemental MediaTailor with Amazon CloudWatch metrics
- Recording AWS Elemental MediaTailor API calls
- Receiving AWS Elemental MediaTailor Channel Assembly alerts
- Tagging AWS Elemental MediaTailor resources
- Monitoring AWS media services with workflow monitor

Viewing AWS Elemental MediaTailor logs

MediaTailor emits logs that describe a variety of milestones and activities in channels and playback configurations. You can use these logs to gain visibility into your workflow and to troubleshoot issues with the service. The following topics describe logs and logging options.

Topics

- AWS Elemental MediaTailor ADS logs description and event types
- AWS Elemental MediaTailor manifest logs description and event types
- AWS Elemental MediaTailor transcode logs description and event types
- Using vended logs to send AWS Elemental MediaTailor logs
- Writing AWS Elemental MediaTailor logs directly to Amazon CloudWatch Logs
- Controlling the volume of AWS Elemental MediaTailor logs
- Filtering AWS Elemental MediaTailor per-session logs and events
- Generating AWS Elemental MediaTailor debug logs

AWS Elemental MediaTailor ADS logs description and event types

The following sections describe the logs that MediaTailor emits to describe events with the ad decision server (ADS). hese are `AdDecisionServerInteractions` logs.

Topics

- AdDecisionServerInteractions events
- ADS log description
- ADS log JSON schema

AdDecisionServerInteractions events

The following events are emitted during MediaTailor interactions with the ad decision server (ADS).

Log	Description
`AD_MARKER_FOUND`	MediaTailor found an ad marker in the manifest.

Log	Description
BEACON_FIRED	MediaTailor fired a tracking beacon.
EMPTY_VAST_RESPONSE	The ADS returned an empty VAST response containing zero ads.
EMPTY_VMAP_RESPONSE	The ADS returned an empty VMAP response.
ERROR_ADS_INVALID_RESPONSE	The ADS returned a non-200 status code.
ERROR_ADS_IO	MediaTailor encountered an error while trying to communicate with the ADS.
ERROR_ADS_RESPONSE_PARSE	MediaTailor encountered an error while parsing the ADS response.
ERROR_ADS_RESPONSE_UNKNOWN_ROOT_ELEMENT	The ADS response contains an invalid root element.
ERROR_ADS_TIMEOUT	The MediaTailor request to the ADS timed out.
ERROR_DISALLOWED_HOST	The ADS host is not allowed.
ERROR_FIRING_BEACON_FAILED	MediaTailor failed at firing the tracking beacon.
ERROR_PERSONALIZATION_DISABLED	Ad insertion is disabled for this session.
ERROR_UNKNOWN	MediaTailor encountered an unknown error during the ADS request.
ERROR_UNKNOWN_HOST	The ADS host is unknown.
ERROR_VAST_INVALID_MEDIA_FILE	The VAST Ad has an invalid or missing `MediaFile` element.
ERROR_VAST_INVALID_VAST_AD_TAG_URI	The VAST response contains an invalid `VASTAdTagURI` .

Log	Description
ERROR_VAST_MISSING_CREATIVES	The VAST Ad contains zero or multiple Creatives elements. Exactly one Creatives element is required.
ERROR_VAST_MISSING_IMPRESSION	The VAST Ad contains zero Impression elements. At least one Impression element is required.
ERROR_VAST_MISSING_MEDIAFILES	The VAST Ad contains zero or multiple MediaFiles elements. Exactly one MediaFiles element is required.
ERROR_VAST_MISSING_OVERLAYS	MediaTailor didn't receive any non-linear creatives from the ad server.
ERROR_VAST_MULTIPLE_LINEAR	The VAST Ad contains multiple Linear elements.
ERROR_VAST_MULTIPLE_TRACKING_EVENTS	The VAST Ad contains multiple TrackingEvents elements.
ERROR_VAST_REDIRECT_EMPTY_RESPONSE	The VAST redirect request returned an empty response.
ERROR_VAST_REDIRECT_FAILED	The VAST redirect request encountered an error.
ERROR_VAST_REDIRECT_MULTIPLE_VAST	The VAST redirect request returned multiple ads.
FILLED_AVAIL	MediaTailor successfully filled the avail.
FILLED_OVERLAY_AVAIL	MediaTailor successfully filled the overlay avail.

Log	Description
INTERSTITIAL_VOD_FAILURE	The ADS request or response encountered a problem while filling interstitial avails for the VOD playlist. No ads will be inserted.
INTERSTITIAL_VOD_SUCCESS	MediaTailor successfully filled interstitial avails for the VOD playlist.
MAKING_ADS_REQUEST	MediaTailor is requesting advertisements from the ADS.
MODIFIED_TARGET_URL	MediaTailor modified the outbound target URL.
NON_AD_MARKER_FOUND	MediaTailor found a non-actionable ad marker in the manifest.
RAW_ADS_RESPONSE	MediaTailor received a raw ADS response.
REDIRECTED_VAST_RESPONSE	MediaTailor received a VAST response after following the VAST redirect.
VAST_REDIRECT	The VAST ad response contains a redirect.
VAST_RESPONSE	MediaTailor received a VAST response.
VOD_TIME_BASED_AVAIL_PLAN_SUCCESS	MediaTailor successfully created a time-based avail plan for the VOD template.
VOD_TIME_BASED_AVAIL_PLAN_VAST_RESPONSE_FOR_OFFSET	MediaTailor is creating a time-based avail plan for the VOD template. MediaTailor received a VAST response for the time offset.
VOD_TIME_BASED_AVAIL_PLAN_WARNING_NO_ADVERTISEMENTS	The ADS request or response encountered a problem while creating a time-based avail plan for the VOD template. No ads will be inserted.

Log	Description
WARNING_NO_ADVERTISEMENTS	MediaTailor encountered a problem while filling the avail. No ads are inserted.
WARNING_URL_VARIABLE_SUBSTITUTION_FAILED	MediaTailor can't substitute dynamic variables in the ADS URL. Check the URL configuration.
WARNING_VPAID_AD_DROPPED	A VPAID ad dropped due to a missing slate, or the session uses server-side reporting.

ADS log description

This section describes the structure and contents of the ADS log description. To explore on your own in a JSON editor, use the listing at the section called "ADS log JSON schema".

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

ADS logs properties

This section describes the properties of the ADS logs.

Property	Type	Required	Description
adsRequestUrl	string	false	The full URL of the ADS request made by MediaTailor.
avail	object of type avail	false	Information about an avail that MediaTail or fills with ads. Currently, for the FILLED_AVAIL event type, this is the plan created by MediaTailor when it first encounters the

Property	Type	Required	Description
			avail. How the avail is eventually filled may vary from this plan, depending on how the content plays out.
`awsAccountId`	string	true	The AWS account ID for the MediaTailor configuration that was used for the session.
`customerId`	string	true	The hashed version of the AWS account ID, which you can use to correlate multiple log entries.
`eventDescription`	string	true	A short description of the event that triggered this log message, provided by the MediaTailor service. By default, this is empty. Example: `Got VAST response`.
`eventTimestamp`	string	true	The date and time of the event.
`eventType`	string	true	The code for the event that triggered this log message. Example: `VAST_RESPONSE` .

Property	Type	Required	Description
originId	string	true	The configuration name from the MediaTailor configuration. This is different from the video content source, which is also part of the configuration.
requestHeaders	array of type requestheaders	false	The headers that MediaTailor included with the ADS request. Typically, the logs include these when a request to the ADS fails, to help with troubleshooting.
requestId	string	true	The MediaTailor request ID, which you can use to correlate multiple log entries for the same request.

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

Property	Type	Required	Description
`vodVastResponseTimeOffset`	number	false	The VMAP specific time offset for VOD ad insertion.

adContent

This section describes the properties of the ADS logs adContent.

ADS Logs adContent Properties

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

adPlaylistUris

This section describes the properties of the ADS logs adPlaylistUris.

ADS Logs adPlaylistUris Properties

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

avail

This section describes the properties of the ADS logs avail.

ADS Logs avail Properties

Property	Type	Required	Description
availId	string	true	The unique identifier for this avail. For HLS, this is the media sequence number where the avail begins. For DASH, this is the period ID.
creativeAds	array of type creativeAd	true	The ads that MediaTailor inserted into the avail.
fillRate	number	true	The rate at which the ads fill the avail duration, from 0.0 (for 0%) to 1.0 (for 100%).
filledDuration	number	true	The sum of the durations of all the ads inserted into the avail.
numAds	number	true	The number of ads that MediaTailor inserted into the avail.
originAvailDuration	number	true	The duration of the avail as specified in the content stream

Property	Type	Required	Description
			from the origin (CUE_OUT or SCTE).
skippedAds	array of type skippedAd	false	The ads that MediaTailor didn't insert, for reasons like TRANSCODE_IN_PROGRESS and TRANSCODE_ERROR .
slateAd	object of type slateAd	true	Information about the slate ad, which MediaTailor uses to fill any unfilled segments in the avail.

creativeAd

This section describes the properties of the ADS logs creativeAd.

ADS Logs creativeAd Properties

Property	Type	Required	Description
adContent	object of type adContent	true	Information about the content of the inserted ad.
creativeUniqueId	string	true	The unique identifier for the ad, used as a key for transcoding. This is the ID field for the creative in the VAST response, if available. Otherwise

Property	Type	Required	Description
			, it's the mezzanine URL of the ad.
`trackingEvents`	object of type trackingEvents	true	The tracking beacon URLs for the various tracking events for the ad. The keys are the event names, and the values are a list of beacon URLs.
`transcodedAdDuration`	number	true	The duration of the ad, calculated from the transcoded asset.
`uri`	string	true	The URL of the mezzanine version of the ad, which is the input to the transcoder.
`vastDuration`	number	true	The duration of the ad, as parsed from the VAST response.

requestheaders

This section describes the properties of the ADS logs requestheaders.

ADS Logs requestheaders Properties

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

Property	Type	Required	Description
value	string	true	The value of the header.

skippedAd

This section describes the properties of the ADS logs skippedAd.

ADS Logs skippedAd Properties

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

Property	Type	Required	Description
vastDuration	number	true	The duration of the ad, as parsed from the VAST response.

slateAd

This section describes the properties of the ADS logs slateAd.

ADS Logs slateAd Properties

Property	Type	Required	Description
adContent	object of type adContent	true	Information about the content of the inserted ad.
creativeUniqueId	string	true	The unique identifier for the ad, used as a key for transcoding. This is the ID field for the creative in the VAST response, if available. Otherwise, it's the mezzanine URL of the ad.
transcodedAdDuration	number	true	The duration of the ad, calculated from the transcoded asset.
uri	string	true	The URL of the mezzanine version of the ad, which is the input to the transcoder.

trackingEvents

This section describes the properties of the ADS logs trackingEvents.

ADS Logs trackingEvents Properties

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

vastAd

This section describes the properties of the ADS logs vastAd.

ADS Logs vastAd Properties

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

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

vastMediaFile

This section describes the properties of the ADS logs vastMediaFile.

ADS Logs vastMediaFile Properties

Property	Type	Required	Description
`apiFramework`	string	true	The API framework needed to manage

Property	Type	Required	Description
			the media file. Example: `VPAID`.
`bitrate`	number	true	The bitrate of the media file.
`delivery`	string	true	The protocol used for the media file, set to either progressive or streaming.
`height`	number	true	The pixel height of the media file.
`id`	string	true	The value of the id attribute of the `MediaFile` tag.
`type`	string	true	The MIME type of the media file, taken from the type attribute of the `MediaFile` tag.
`uri`	string	true	The URL of the mezzanine version of the ad, which is the input to the transcoder.
`width`	number	true	The pixel width of the media file.

vastResponse

This section describes the properties of the ADS logs vastResponse.

ADS Logs vastResponse Properties

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

vodCreativeOffsets

This section describes the properties of the ADS logs vodCreativeOffsets.

ADS Logs vodCreativeOffsets Properties

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

vodCreativeOffset

This section describes the properties of the ADS logs vodCreativeOffset.

ADS Logs vodCreativeOffset Properties

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

Property	Type	Required	Description
vastDuration	number	true	The duration of the ad, as parsed from the VAST response.

ADS log JSON schema

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

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://amazon.com/elemental/midas/mms/adsLogSchema.json",
  "type": "object",
  "title": "AWS Elemental MediaTailor ADS Log JSON Schema",
  "required": [
    "eventType",
    "eventTimestamp",
    "requestId",
    "sessionType",
    "eventDescription",
    "awsAccountId",
    "customerId",
    "originId",
    "sessionId"
  ],
  "additionalProperties": false,
  "properties": {
    "eventType": {
      "$id": "#/properties/eventType",
      "type": "string",
      "description": "The code for the event that triggered this log message. Example:
 <code>VAST_RESPONSE</code>.",
      "examples": [
        "FILLED_AVAIL"
      ]
    },
    "eventTimestamp": {
      "$id": "#/properties/eventTimestamp",
      "type": "string",
      "description": "The date and time of the event.",
```

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

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

```
      "additionalProperties":  false,
      "properties": {
        "originAvailDuration": {
          "$id": "#/properties/avail/originAvailDuration",
          "type": "number",
          "description": "The duration of the avail as specified in the content stream
from the origin (<code>CUE_OUT</code> or <code>SCTE</code>)."
        },
        "filledDuration": {
          "$id": "#/properties/avail/filledDuration",
          "type": "number",
          "description": "The sum of the durations of all the ads inserted into the
avail."
        },
        "fillRate": {
          "$id": "#/properties/avail/fillRate",
          "type": "number",
          "description": "The rate at which the ads fill the avail duration, from 0.0
(for 0%) to 1.0 (for 100%)."
        },
        "driftMillisecondsAtAvailStart": {
          "$id": "#/properties/avail/driftMillisecondsAtAvailStart",
          "type": "number",
          "description": "The cumulative drift at the beginning of this avail. A
positive value implies that we are moving away from the live edge, a negative value
implies that we are moving towards the live edge."
        },
        "creativeAds": {
          "$id": "#/properties/avail/creativeAds",
          "type": "array",
          "description": "The ads that MediaTailor inserted into the avail.",
          "items": {
            "type": "object",
            "title": "creativeAd",
            "description": "Information about a single inserted ad.",
            "required": [
              "uri",
              "creativeUniqueId",
              "adSystem",
              "adContent",
              "trackingEvents",
              "vastDuration",
              "transcodedAdDuration"
            ],
```

```
            "additionalProperties": false,
            "properties": {
              "uri": { "$ref": "#/definitions/adMezzanineUri" },
              "creativeUniqueId": { "$ref": "#/definitions/creativeUniqueId" },
              "adSystem": { "$ref": "#/definitions/adSystem" },
              "adContent": { "$ref": "#/definitions/adContent" },
              "trackingEvents": { "$ref": "#/definitions/trackingEvents" },
              "vastDuration": { "$ref": "#/definitions/vastDuration" },
              "transcodedAdDuration": { "$ref": "#/definitions/transcodedAdDuration" }
            }
          }
        },
        "numAds": {
          "$id": "#/properties/avail/numAds",
          "type": "number",
          "description": "The number of ads that MediaTailor inserted into the avail."
        },
        "slateAd": {
          "$id": "#/properties/avail/slateAd",
          "type": ["object", "null"],
          "title": "slateAd",
          "description": "Information about the slate ad, which MediaTailor uses to
fill any unfilled segments in the avail.",
          "additionalProperties": false,
          "required": [
            "uri",
            "creativeUniqueId",
            "adContent",
            "transcodedAdDuration"
          ],
          "properties": {
            "uri": { "$ref": "#/definitions/adMezzanineUri" },
            "creativeUniqueId": { "$ref": "#/definitions/creativeUniqueId" },
            "adContent": { "$ref": "#/definitions/adContent" },
            "transcodedAdDuration": { "$ref": "#/definitions/transcodedAdDuration" }
          }
        },
        "availId": {
          "$id": "#/properties/avail/availId",
          "type": "string",
          "description": "The unique identifier for this avail. For HLS, this is the
media sequence number where the avail begins. For DASH, this is the period ID."
        },
        "skippedAds": {
```

```
          "$id": "#/properties/avail/skippedAds",
          "type": "array",
          "description": "The ads that MediaTailor didn't insert, for reasons like
<code>TRANSCODE_IN_PROGRESS</code> and <code>TRANSCODE_ERROR</code>.",
          "items": {
            "type": "object",
            "title": "skippedAd",
            "description": "Information about a single skipped ad.",
            "required": [
              "creativeUniqueId",
              "adMezzanineUrl",
              "skippedReason",
              "vastDuration"
            ],
            "additionalProperties": false,
            "properties": {
              "creativeUniqueId": { "$ref": "#/definitions/creativeUniqueId" },
              "adMezzanineUrl": {
                "type": "string",
                "description": "The mezzanine URL of the skipped ad."
              },
              "skippedReason": {
                "type": "string",
                "description": "The code that indicates why the ad wasn't inserted.
Example: <code>TRANSCODE_IN_PROGRESS</code>."
              },
              "vastDuration": { "$ref": "#/definitions/vastDuration" },
              "transcodedAdDuration": { "$ref": "#/definitions/transcodedAdDuration" },
              "targetVariant": {
                "type": "object",
                "title": "targetVariant",
                "description": "The target variant of the source content. This key is
present when an ad wasn't inserted due to the source content containing a variant that
could not match to any variants present in this ad.",
                "required": [
                  "mediaProtocol",
                  "mediaType",
                  "bitrate",
                  "mediaResolution",
                  "codecs"
                ],
                "additionalProperties": false,
                "properties": {
                  "mediaProtocol": {
```

```
                    "type": "string",
                    "description": "The media protocol of this variant, such as HLS.",
                    "enum": [
                      "HLS",
                      "DASH"
                    ]
                  },
                  "mediaType": {
                    "type": "array",
                    "description": "The media type of this variant, such as VIDEO.",
                    "items": {
                      "type": "string",
                      "enum": [
                        "VIDEO",
                        "AUDIO",
                        "SUBTITLES",
                        "TRICK_PLAY"
                      ],
                      "description": "Media type, such as VIDEO."
                    }
                  },
                  "bitrate": {
                    "$ref": "#/definitions/bitrate"
                  },
                  "mediaResolution": {
                    "type": "object",
                    "title": "mediaResolution",
                    "description": "The media resolution of this variant.",
                    "required": [
                      "width",
                      "height"
                    ],
                    "additionalProperties": false,
                    "properties": {
                      "width": {
                        "$ref": "#/definitions/width"
                      },
                      "height": {
                        "$ref": "#/definitions/height"
                      }
                    }
                  },
                  "codecs": {
                    "type": "array",
```

```
                    "description": "The codecs of this variant.",
                    "items": {
                      "type": "string",
                      "description": "Codec, such as avc1."
                    }
                  }
                }
              }
            }
          }
        },
        "adBreakTrackingEvents": {
          "$id": "#properties/avail/adBreakTrackingEvents",
          "type": "object",
          "title": "adBreakTrackingEvents",
          "description": "VMAP/ad break tracking events and corresponding URL",
          "properties": {
            "items": {
              "$ref": "#/definitions/adBreakTrackingEvents"
            }
          }
        }
      }
    },

    "vastResponse": {
      "$id": "#/properties/vastResponse",
      "type": "object",
      "title": "vastResponse",
      "description": "Information about the VAST response that MediaTailor received
from the ADS.",
      "required": [
        "version",
        "vastAds",
        "errors",
        "nonLinearAdsList"
      ],
      "additionalProperties":  false,
      "properties": {
        "version": {
          "$id": "#/properties/vastResponse/version",
          "type": "string",
          "description": "The VAST specification version, parsed from the
<code>version</code> attribute of the <code>VAST</code> tag in the response.",
```

```
          "examples": [
            "3.0"
          ],
          "pattern": "^(.*)$"
        },
        "vastAds": {
          "$id": "#/properties/vastResponse/vastAds",
          "type": "array",
          "description": "The ads parsed from the VAST response.",
          "items": {
            "$ref": "#/definitions/vastAd"
          }
        },
        "errors": {
          "$id": "#/properties/vastResponse/errors",
          "type": "array",
          "description": "The error URLs parsed from the <code>Error</code> tags in the
VAST response.",
          "items": {
            "type": "string",
            "description": "A single error URL."
          }
        },
        "nonLinearAdsList": {
          "$id": "#/properties/vastResponse/nonLinearAds",
          "type": "array",
          "description": "A list of NonLinearAds as they are read from the VAST
response.",
          "items": {
            "$ref": "#/definitions/nonLinearAds"
          }
        }
      }
    },

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

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

```
        5.0
      ]
    },

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

    "adsRequestUrl": {
      "$id": "#/properties/adsRequestUrl",
      "type": "string",
```

```
      "description": "The full URL of the ADS request made by MediaTailor."
    },
    "adMarkers": {
      "$id": "#/properties/adMarkers",
      "type": "string",
      "description": "Found Ad Marker in the Manifest."
    },
    "segmentationUpid": {
      "$id": "#/properties/segmentationUpid",
      "type": "string",
      "description": "Value of segmentation upid parsed from ad markers in manifest."
    },
    "segmentationTypeId": {
      "$id": "#/properties/segmentationTypeId",
      "type": "integer",
      "description": "Value of segmentation typeId parsed from ad markers in manifest."
    },
    "requestHeaders": {
      "$id": "#/properties/requestHeaders",
      "type": "array",
      "description": "The headers that MediaTailor included with the ADS request.
Typically, the logs include these when a request to the ADS fails, to help with
troubleshooting.",
      "items": {
        "type": "object",
        "title": "requestheaders",
        "description": "The name and value for a single header included in the ADS
request.",
        "required": [
          "name",
          "value"
        ],
        "additionalProperties": false,
        "properties": {
          "name": {
            "type": "string",
            "description": "The name of the header."
          },
          "value": {
            "type": "string",
            "description": "The value of the header."
          }
        }
      }
```

```
    },

    "originalTargetUrl": {
      "$id": "#/properties/originalTargetUrl",
      "type": "string",
      "description": "The old URL to which MediaTailor was going to make a request."
    },
    "updatedTargetUrl": {
      "$id": "#/properties/updatedTargetUrl",
      "type": "string",
      "description": "The new URL to which MediaTailor is making a request."
    },

    "rawAdsResponse": {
      "$id": "#/properties/rawAdsResponse",
      "type": "string",
      "description": "Paginated ADS response as it's exactly returned to MediaTailor."
    },
    "rawAdsResponseIndex": {
      "$id": "#/properties/rawAdsResponseIndex",
      "type": "integer",
      "description": "Integer value denoting this rawAdsResponse's index into the
full ADS response. This value is used to order the paginated messages for this ADS
response."
    }
  },

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

  "oneOf": [
    { "$ref": "#/definitions/eventAdMarkersFound" },
    { "$ref": "#/definitions/eventNonAdMarkerFound" },
    { "$ref": "#/definitions/eventMakingAdsRequest" },
    { "$ref": "#/definitions/eventModifiedTargetUrl" },
    { "$ref": "#/definitions/eventRawAdsResponse" },
    { "$ref": "#/definitions/eventVastResponse" },
    { "$ref": "#/definitions/eventFilledAvail" },
    { "$ref": "#/definitions/eventFilledOverlayAvail" },
    { "$ref": "#/definitions/eventErrorFiringBeaconFailed" },
    { "$ref": "#/definitions/eventWarningNoAdvertisements" },
    { "$ref": "#/definitions/eventUnknownHost" },
    { "$ref": "#/definitions/eventErrorAdsTimeout" },
```

```
    { "$ref": "#/definitions/eventErrorVastMissingOverlays" },
    { "$ref": "#/definitions/eventPlannedAvail" },
    { "$ref": "#/definitions/eventEmptyVastResponse" },
    { "$ref": "#/definitions/eventEmptyVmapResponse" },
    { "$ref": "#/definitions/eventErrorUnknown" },
    { "$ref": "#/definitions/eventVastRedirect" },
    { "$ref": "#/definitions/eventRedirectedVastResponse" },
    { "$ref": "#/definitions/eventErrorAdsMissingImpression" },
    { "$ref": "#/definitions/eventErrorAdsResponseParse" },
    { "$ref": "#/definitions/eventErrorAdsInvalidResponse" },
    { "$ref": "#/definitions/eventErrorDisallowedHost"},
    { "$ref": "#/definitions/eventPersonalizationDisabled"},
    { "$ref": "#/definitions/eventWarningDynamicVariableSubFailed"},
    { "$ref": "#/definitions/eventVodTimeBasedAvailPlanVastResponseForOffset" },
    { "$ref": "#/definitions/eventVodTimeBasedAvailPlanSuccess" }
  ],

  "definitions": {
    "eventAdMarkersFound": {
      "$id": "#/definitions/eventAdMarkersFound",
      "required": [
        "eventType",
        "adMarkers"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "AD_MARKER_FOUND"
        }
      }
    },
    "eventNonAdMarkerFound": {
      "$id": "#/definitions/eventNonAdMarkerFound",
      "required": [
        "eventType",
        "adMarkers"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "NON_AD_MARKER_FOUND"
        }
      }
    },
```

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

    "eventModifiedTargetUrl": {
      "$id": "#/definitions/eventModifiedTargetUrl",
      "required": [
        "eventType",
        "originalTargetUrl",
        "updatedTargetUrl"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "MODIFIED_TARGET_URL"
        }
      }
    },

    "eventRawAdsResponse": {
      "$id": "#/definitions/eventRawAdsResponse",
      "required": [
        "eventType",
        "rawAdsResponse",
        "rawAdsResponseIndex"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "RAW_ADS_RESPONSE"
        }
      }
    },
```

```
    "eventVastResponse": {
      "$id": "#/definitions/eventVastResponse",
      "_comment": "NOTE: the vastResponse property should ideally be marked as a
required field for this event, but unfortunately, in the case of an empty vast
response, we currently emit an EMPTY_VAST_RESPONSE followed by a VAST_RESPONSE, and
the vastResponse property is not present in the latter. We need to fix this so that we
don't emit both of those events in the empty response case, and update this schema to
flag vastResponse as required for VAST_RESPONSE.",
      "required": [
        "eventType"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "VAST_RESPONSE"
        }
      }
    },

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

    "eventFilledOverlayAvail": {
      "$id": "#/definitions/eventFilledOverlayAvail",
      "required": [
        "eventType",
        "avail"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "FILLED_OVERLAY_AVAIL"
        }
```

```
      }
    },

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

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

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

```
        }
      }
    },

    "eventUnknownHost": {
      "$id": "#/definitions/eventUnknownHost",
      "required": [
        "eventType",
        "requestHeaders"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "ERROR_UNKNOWN_HOST"
        }
      }
    },

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

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

```
      }
    },

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

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

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

```
    "eventVastRedirect": {
      "$id": "#/definitions/eventVastRedirect",
      "required": [
        "eventType"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "VAST_REDIRECT"
        }
      }
    },

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

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

    "eventErrorAdsInvalidResponse": {
      "$id": "#/definitions/eventErrorAdsInvalidResponse",
```

```
      "required": [
        "eventType",
        "additionalInfo"
      ],
      "properties": {
        "eventType": {
          "type": "string",
          "const": "ERROR_ADS_INVALID_RESPONSE"
        }
      }
    },

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

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

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

```
        "eventType": {
          "type": "string",
          "const": "ERROR_PERSONALIZATION_DISABLED"
        }
      }
    },

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

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

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

```
          "const": "VOD_TIME_BASED_AVAIL_PLAN_SUCCESS"
        }
      }
    },

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

    "adSystem": {
      "type": "string",
      "description": "The value of the <code>AdSystem</code> tag in the VAST response.
"
    },

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

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

    "adContent": {
      "$id": "#/properties/adContent",
      "type": ["object", "null"],
      "title": "adContent",
      "description": "Information about the content of the inserted ad.",
      "additionalProperties": false,
      "properties": {
        "adPlaylistUris": {
          "$id": "#/properties/adContent/adPlaylistUris",
          "type": "object",
          "title": "adPlaylistUris",
          "description": "The mapping from the origin manifest for a variant to the ad
manifest for the variant. For DASH, this contains a single entry, because all variants
are represented in a single DASH manifest. ",
          "additionalProperties": {
            "$id": "#/properties/adContent/adPlaylistUris/adPlaylistUri",
```

```
            "type": "string",
            "description": "The URL of the ad manifest for the specific variant."
          }
        }
      }
    },

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

    "bitrate": {
      "type": "integer",
      "examples": [
        533
      ],
      "description": "The bitrate."
    },
    "width": {
      "type": "integer",
      "examples": [
        1280
      ],
      "description": "Width in pixels."
    },
    "height": {
      "type": "integer",
      "examples": [
        720
      ],
      "description": "Height in pixels."
    },

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

      "additionalProperties":  {
        "type": "array",
```

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

    "nonLinearAds": {
      "$id": "#/properties/nonLinearAds",
      "type": "object",
      "title": "nonLinearAds",
      "description": "A NonLinearAds as it appears in the VAST response.",
      "required": [
        "nonLinearAdList",
        "nonLinearTrackingEvents"
      ],
      "properties": {
        "nonLinearAdList": {
          "type": "array",
          "description": "List of non linear ads as they exist within one
NonLinearAds.",
          "items": {
            "type": "object",
            "title": "nonLinearAdList",
            "description": "List of NonLinearAd as they are parsed from its parent
NonLinearAds.",
            "properties": {
              "nonLinearAdId": {
                "type": "string",
                "description": "Ad ID of this non linear ad."
              },
              "nonLinearAdSystem": {
                "type": "string",
                "description": "Ad system of this non linear ad's parent Inline ad."
              },
              "nonLinearAdTitle": {
                "type": "string",
                "description": "Ad title of this non linear ad's parent Inline ad."
              },
              "nonLinearCreativeId": {
                "type": "string",
```

```
                "description": "Creative ID of this non linear ad's parent Creative
ad."
              },
              "nonLinearCreativeAdId": {
                "type": "string",
                "description": "Creative ad ID of this non linear ad."
              },
              "nonLinearCreativeSequence": {
                "type": "string",
                "description": "Creative sequence of this non linear ad."
              },
              "nonLinearWidth": {
                "type": "string",
                "description": "Width of this non linear ad."
              },
              "nonLinearHeight": {
                "type": "string",
                "description": "Height of this non linear ad."
              },
              "nonLinearExpandedWidth": {
                "type": "string",
                "description": "Expanded width of this non linear ad."
              },
              "nonLinearExpandedHeight": {
                "type": "string",
                "description": "Expanded height of this non linear ad."
              },
              "nonLinearScalable": {
                "type": "boolean",
                "description": "Boolean denoting if this non linear ad is scalable."
              },
              "nonLinearMaintainAspectRatio": {
                "type": "boolean",
                "description": "Boolean denoting if aspect ratio should be maintained
for this non linear ad."
              },
              "nonLinearMinSuggestedDuration": {
                "type": "string",
                "description": "Min suggested duration for this non linear ad."
              },
              "nonLinearApiFramework": {
                "type": "string",
                "description": "API framework for this non linear ad's parent Inline
ad."
```

```
            },
            "nonLinearStaticResource": {
              "type": "string",
              "description": "Static resource for this non linear ad."
            },
            "nonLinearStaticResourceCreativeType": {
              "type": "string",
              "description": "Static Resource creative type for this non linear ad."
            },
            "nonLinearIFrameResource": {
              "type": "string",
              "description": "I-Frame resource for this non linear ad."
            },
            "nonLinearHtmlResource": {
              "type": "string",
              "description": "HTML resource for this non linear ad."
            },
            "nonLinearAdParameters": {
              "type": "string",
              "description": "Ad parameters for this non linear ad."
            },
            "nonLinearClickThrough": {
              "type": "string",
              "description": "Click Through data for this non linear ad."
            },
            "nonLinearClickTracking": {
              "type": "string",
              "description": "Click Tracking data for this non linear ad."
            },
            "nonLinearClickTrackingId": {
              "type": "string",
              "description": "Click Tracking ID for this non linear ad."
            }
          }
        }
      },
      "nonLinearTrackingEvents": { "$ref": "#/definitions/trackingEvents" },
      "extensions": {
        "$id": "#/properties/nonLinearAds/extensions",
        "type": "array",
        "description": "Extensions that exist for this NonLinearAds.",
        "items": {
          "$id": "#/properties/nonLinearAds/extensions/items",
          "type": "object",
```

```
            "title": "Extensions",
            "description": "Extensions found in non linear ads",
            "additionalProperties":  false,
            "properties": {
              "extensionType": {
                "$id": "#/properties/nonLinearAds/extensions/extensionType",
                "type": "string",
                "description": "The value of the extension type attribute of the
<code>Extensions</code> tag.",
                "examples": [
                  "FreeWheel"
                ]
              },
              "extensionContent": {
                "$id": "#/properties/nonLinearAds/extensions/extensionContent",
                "type": "string",
                "description": "The extension content attribute of the
<code>Extensions</code> tag.",
                "examples": [
                  "progressive"
                ]
              }
            }
          }
        }
      }
    },
    "adBreakTrackingEvents": {
      "$id": "#/properites/adBreakTrackingEvents",
      "type": "object",
      "title": "adBreakTrackingEvents",
      "description": "These are all VMAP ad break tracking events.",
      "additionalProperties": {
        "type": "array",
        "description": "VMAP/ad break tracking events and corresponding URL",
        "items": {
          "type": "string",
          "description": "The beacon URLs for this tracking event."
        }
      }
    },
    "vastAd": {
      "$id": "#/properties/vastAd",
      "type": "object",
```

```
      "title": "vastAd",
      "description": "Information about a single ad parsed from the VAST response.",
      "required": [
        "vastAdId",
        "adSystem",
        "adTitle",
        "creativeId",
        "creativeAdId",
        "duration",
        "vastMediaFiles",
        "trackingEvents"
      ],
      "additionalProperties":  false,
      "properties": {
        "vastAdId": {
          "$id": "#/properties/vastAd/vastAdId",
          "type": "string",
          "description": "The value of the id attribute of the <code>Ad</code> tag in
the VAST response",
          "examples": [
            "ad1"
          ]
        },
        "adSystem": {"$ref": "#/definitions/adSystem" } ,
        "adTitle": {
          "$id": "#/properties/vastAd/adTitle",
          "type": "string",
          "description": "The media files that are available for the ad in the VAST
response.",
          "examples": [
            "External NCA1C1L1 LinearInlineSkippable"
          ]
        },
        "creativeId": {
          "$id": "#/properties/vastAd/creativeId",
          "type": "string",
          "description": "The value of the id attribute of the <code>Creative</code>
tag in the VAST response.",
          "examples": [
            "creative1"
          ]
        },
        "creativeAdId": {
          "$id": "#/properties/vastAd/creativeAdId",
```

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

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

```
          "examples": [
            "https://ads.redirect.com/redirect1"
          ]
        }
      }
    }
  }
}
```

AWS Elemental MediaTailor manifest logs description and event types

The following sections describe the logs that MediaTailor emits to describe events with the origin server when requesting and receiving a manifest. These are `ManifestService` logs.

Topics

- ManifestService events
- Manifest logs properties

ManifestService events

The following events are emitted during MediaTailor interactions with the origin.

Log	Description
`CONFIG_SECURITY_ERROR`	The MediaTailor configuration has a security issue.
`CONFIG_SYNTAX_ERROR`	The origin and asset path result in a malformed URL.
`CONNECTION_ERROR`	The MediaTailor connection to the origin was refused or failed.
`GENERATED_MANIFEST`	MediaTailor generated a manifest.
`HOST_DISALLOWED`	MediaTailor does not allow HTTP requests to this host.

Log	Description
INCOMPATIBLE_HLS_VERSION	The manifest uses an incompatible HLS version. MediaTailor requires version 3 or greater.
INVALID_SINGLE_PERIOD_DASH_MANIFEST	The single-period DASH manifest is invalid. MediaTailor is passing through single-period DASH manifest.
IO_ERROR	MediaTailor encountered an IO error during communication with the origin.
LAST_PERIOD_MISSING_AUDIO	The last period in the DASH manifest is missing all audio `AdaptationSets` because of origin audio or video misalignment. To avoid playback issues, delay publishing the last period until at least the next request.
LAST_PERIOD_MISSING_AUDIO_WARNING	The last period in the DASH manifest is missing all audio `AdaptationSets` because of origin audio or video misalignment. Choosing to publish (not delay) the last period. Missing audio might cause playback issues.
MANIFEST_ERROR	The MediaTailor manifest request failed.
NO_MASTER_OR_MEDIA_PLAYLIST	The origin response doesn't contain a primary playlist or media playlist.
NO_MASTER_PLAYLIST	The origin response doesn't contain the expected primary playlist.
NO_MEDIA_PLAYLIST	The origin response doesn't contain the expected media playlist.
ORIGIN_MANIFEST	MediaTailor fetched an origin manifest.

Log	Description
PARSING_ERROR	The origin is unable to parse the manifest request.
SCTE35_PARSING_ERROR	MediaTailor is unable to parse `Signal Binary` element in the manifest.
SESSION_INITIALIZED	A session was initialized.
TIMEOUT_ERROR	The MediaTailor manifest request timed out.
TRACKING_RESPONSE	MediaTailor served a tracking response.
UNKNOWN_ERROR	MediaTailor encountered an unknown error.
UNKNOWN_HOST	The host is unknown.
UNSUPPORTED_SINGLE_PERIOD_DASH_MANIFEST	The single-period DASH manifest is unsupported. MediaTailor is passing through single-period DASH manifest.

Manifest logs properties

This section describes the properties of the manifest logs.

Property	Type	Required
awsAccountId	string	true
eventTimestamp	string	true
originId	string	true
customerId	string	false
eventType	string	false
sessionId	string	false

Property	Type	Required
originRequestUrl	string	false
mediaTailorPath	string	false
requestId	string	false
responseBody	string	false
sessionType	string (legal values: [DASH, HLS])	false
requestNextToken	string	false
eventDescription	string	false
assetPath	string	false
originFullUrl	string	false
originPrefixUrl	string	false
additionalInfo	string	false
cause	string	false
response	string	false
httpCode	string	false
errorMessage	string	false
adAdsResponse	string	false
adAdsRawResponse	string	false
adAdsRequest	string	false
adNumNewAvails	string	false

Property	Type	Required
generatedMediaPlaylist	string	false
	string	false
requestStartTime	string	false
requestEndTime	string	false
requestStartTimeEpochMillis	string	false
requestEndTimeEpochMillis	string	false

AWS Elemental MediaTailor transcode logs description and event types

The following sections describe the logs that MediaTailor emits to describe events with the transcode service when preparing creatives for ad stitching. These are `TranscodeService` logs.

Topics

- TranscodeService events
- Transcode logs properties

TranscodeService events

The following events are emitted during MediaTailor interactions while transcoding ads.

Log	Description
IMPORT_ERROR	MediaTailor encountered an internal error during an import job (for preconditioned ads). Using an empty set of ads.

Log	Description
INITIALIZED	MediaTailor initialized either a transcode job or an import job (for preconditioned ads).
INTERNAL_ERROR	MediaTailor encountered an internal error. Using an empty set of ads.
MISSING_VARIANTS	MediaTailor could not transcode the ad because of missing variants. Using an empty set of ads.
PROFILE_NOT_FOUND	MediaTailor could not transcode the ad because of a missing profile to transcode. Using an empty set of ads.
TRANSCODE_COMPLETED	Video transcoding is complete. The ad can be used for ad insertion.
TRANSCODE_ERROR	MediaTailor encountered an internal error during a transcode job. Using an empty set of ads.
TRANSCODE_IN_PROGRESS	Video transcoding is in progress. The transcoded video is not ready. Using an empty set of ads.

Transcode logs properties

This section describes the properties of the transcode logs.

Property	Type	Required	Description
awsAccountId	string	true	The AWS account ID for the MediaTailor configuration that was used for the session.

Property	Type	Required	Description
`eventTimestamp`	string	true	The date and time of the event.
`originId`	string	true	The configuration name from the MediaTailor configuration. This is different from the video content source, which is also part of the configuration.
`eventType`	string	false	The code for the event that triggered this log message. Example: `TRANSCODE_ERROR` .
`eventDescription`	string	false	A short description of the event that triggered this log message, provided by the MediaTailor service. By default, this is empty.

Property	Type	Required	Description
sessionId	string	false	The unique numeric identifier that MediaTailor assigned to the player session. All requests that a player makes for a session have the same session ID. Example: e039fd39-09f0-46b2-aca9-9871cc116cde .
creativeUniqueId	string	false	The unique identifier for the ad creative that's being transcoded.
profileName	string	false	
adUri	string	false	The URI for the ad creative.
transcodeRequestId	string	false	The unique identifier for this transcode request.
cacheStatus	string	false	Indicates if MediaTail or cached the transcoded ad.

Using vended logs to send AWS Elemental MediaTailor logs

You can use vended logs for greater flexibility and control over where to deliver logs that MediaTailor emits from your playback configuration.

With vended logs, MediaTailor sends all log activity associated with a configuration to Amazon CloudWatch Logs. CloudWatch Logs then sends the percent of logs that you specify to your chosen destination. Supported destinations are an Amazon CloudWatch Logs log group, Amazon S3 bucket, or Amazon Data Firehose stream.

Because vended logs are available at volume discount pricing, you could see cost savings compared to sending logs directly to CloudWatch Logs. For pricing, see *Vended Logs* on the **Logs** tab at Amazon CloudWatch Pricing.

To use vended logs, you must do the following:

1. Add permissions.
2. Create log delivery destinations.
3. Configure log delivery in CloudWatch Logs.
4. Enable vended logs in MediaTailor.

For more information about vended logs, see Enable logging from AWS services in the CloudWatch Logs user guide. MediaTailor supports V2 of vended logs.

Step 1: Add permissions for MediaTailor log delivery

The person who's setting up vended logs must have permissions to create the delivery destination, configure log delivery, and enable vended logs in MediaTailor. Use the following policies to ensure that you have the appropriate permissions to set up vended logs.

Policies for CloudWatch Logs and delivery destinations

The following sections in the *Amazon CloudWatch Logs User Guide* provide the policies that enable you to work with logs in CloudWatch Logs and your delivery destinations. If you send logs to multiple locations, you can combine the policy statements into one policy instead of creating multiple policies.

- Logs sent to CloudWatch Logs
- Logs sent to Amazon S3
- Logs sent to Firehose

Policy for set up from the console

If you're setting up vended logs delivery through the console instead of the API or AWS CLI, you must have the following additional permissions in your policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLogDeliveryActionsConsoleCWL",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleS3",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleFH",
            "Effect": "Allow",
            "Action": [
                "firehose:ListDeliveryStreams",
                "firehose:DescribeDeliveryStream"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Policy for vended logs in MediaTailor

To create, view, or modify vended logs delivery in MediaTailor, you must have the following permissions in your policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ServiceLevelAccessForLogDelivery",
            "Effect": "Allow",
            "Action": [
                "mediatailor:AllowVendedLogDeliveryForResource"],
            "Resource":
 "arn:aws:mediatailor:region:111122223333:playbackConfiguration/*"
        }
    ]
}
```

For information about adding permissions and working with policies, see Identity and Access Management for AWS Elemental MediaTailor.

Step 2: Create delivery destinations for MediaTailor logs

Create the resources where your logs will be sent. Record the ARN of the resource for use in configuring the log delivery in a later step.

CloudWatch Logs log group delivery destination

Use one of the following for help creating a log group.

- For the console, see Create a log group in CloudWatch Logs in the *Amazon CloudWatch Logs User Guide*.
- For the API, see CreateLogGroup in the *Amazon CloudWatch Logs API Reference.*
- For SDKs and CLI, see Use `CreateLogGroup` with an AWS SDK or AWS CLI in the *Amazon CloudWatch Logs User Guide*.

Amazon S3 bucket delivery destination

Use one of the following for help creating a bucket.

- For the console, SDKs, and CLI, see Create a bucket in the *Amazon Simple Storage Service User Guide*.
- For the API, see CreateBucket in the *Amazon Simple Storage Service API Reference*.

Firehose stream delivery destination

For help creating a stream, see Create a Firehose stream from console in the *Amazon Data Firehose Developer Guide*.

Step 3: Enable vended logs for the MediaTailor playback configuration

Create or update the playback configuration that will be sending logs to the delivery destination that you created in the previous step. Record the name of the configuration for use in configuring the log delivery in a later step.

- To enable vended logs through the console, using Creating a configuration or Editing a configuration Editing a configuration to access the **Logging** settings. For **Logging strategies**, choose **Vended logs**.
- To enable vended logs through the API, you must have an existing configuration. Use `ConfigureLogsForPlaybackConfiguration` to add the logging strategy `Vended logs`.

If you're using the legacy MediaTailor logging strategy of sending logs directly to CloudWatch Logs and want to migrate to vended logs, see Migrating the logging strategy.

> **Important**
>
> If you change the log strategy from Legacy CloudWatch to vended logs, MediaTailor will make this change as soon as you save the updates. You will stop receiving logs until you have fully configured vended logging.

Step 4: Configure log delivery in CloudWatch Logs

In CloudWatch Logs, you must create three elements to represent the pieces of log delivery. These elements are described in detail in CreateDelivery in the *Amazon CloudWatch Logs API Reference*. The high-level steps to configure the delivery with the CloudWatch Logs API are as follows.

To configure log delivery in CloudWatch Logs (API)

1. Use `PutDeliverySource` to add the source of logs.

   A `DeliverySource` represents the playback configuration that's generating the logs. You need the name of the playback configuration to create the `DeliverySource`.

2. Use PutDeliveryDestination to add the destination where logs will be written.

   A DeliveryDestination represents the delivery destination. You need the ARN of the log group, bucket, or stream to create the DeliveryDestination.

3. Use PutDeliveryDestinationPolicy if you are delivering logs across accounts.

   If the delivery destination is in a different account from the playback configuration, you need a DeliveryDestinationPolicy. This policy allows CloudWatch Logs to deliver logs to the DeliveryDestination.

4. Use CreateDelivery to link the DeliverySource to the DeliveryDestination.

   A Delivery represents the connection between the DeliverySource and DeliveryDestination.

Migrating your AWS Elemental MediaTailor logging strategy

If you change the log strategy from Legacy CloudWatch to vended logs, MediaTailor will make this change as soon as you save the updates. To avoid interruptions in your logging workflow, use the following steps to migrate your logging strategy.

1. Follow the steps as described in Using vended logs. For Enable vended logs in MediaTailor, select *both* logging strategies (**Vended logs** and **Legacy CloudWatch**).

   MediaTailor will send logs through both vended logs and directly to CloudWatch Logs.

2. Make the necessary changes in your workflow that are dependent on your logging strategy and delivery destination.
3. Revisit Enable vended logs in MediaTailor and remove **Legacy CloudWatch** from the **Logging strategies**.

Writing AWS Elemental MediaTailor logs directly to Amazon CloudWatch Logs

MediaTailor produces logs that contain detailed information about session activity and ad decision server interactions, and writes them to Amazon CloudWatch. The logs provide a sequential description of activity that occurs during the session.

MediaTailor can also use vended logs for flexibility in log delivery and volume discount pricing. For information about vended logs, see Using vended logs.

Topics

- Permissions for Amazon CloudWatch Logs
- "As Run" log for AWS Elemental MediaTailor Channel Assembly
- AWS Elemental MediaTailor ADS log analysis in Amazon CloudWatch Logs Insights

Permissions for Amazon CloudWatch Logs

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

To allow MediaTailor access to CloudWatch

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
3. Choose the **Another AWS account** role type.
4. For **Account ID**, enter your AWS account ID.
5. Select **Require external ID** and enter **`Midas`**. This option automatically adds a condition to the trust policy that allows the service to assume the role only if the request includes the correct `sts:ExternalID`.
6. Choose **Next: Permissions**.
7. Add a permissions policy that specifies what actions this role can complete. Select from one of the following options, and then choose **Next: Review**:
   - **CloudWatchLogsFullAccess** to provide full access to Amazon CloudWatch Logs
   - **CloudWatchFullAccess** to provide full access to Amazon CloudWatch
8. For **Role name**, enter **`MediaTailorLogger`**, and then choose **Create role**.
9. On the **Roles** page, choose the role that you just created.
10. To update the principal, edit the trust relationship:
    1. On the role's **Summary** page, choose the **Trust relationship** tab.
    2. Choose **Edit trust relationship**.

3. In the policy document, change the principal to the MediaTailor service. It should look like this:

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

   The entire policy should read as follows:

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

4. Choose **Update Trust Policy**.

"As Run" log for AWS Elemental MediaTailor Channel Assembly

The *As Run* log, in the CloudWatch `MediaTailor/Channel/AsRunLog` log group, shows information about programs and ad breaks as they play.

When you create a channel, the As Run log is disabled by default. Using the Console or the AWS Command Line Interface (AWS CLI), you can enable and disable the As Run log state for each channel in your account.

When you enable the As Run log, MediaTailor automatically creates a service-linked role that allows MediaTailor to write and manage the As Run log in your CloudWatch Logs account. For more information about service-linked roles, see Using service-linked roles for MediaTailor.

> **Note**
>
> The As Run Log currently only supports the default program. For now it doesn't support the alternateMedia created by program rules. This means that it currently does not generate the As Run Log for alternateMedia.

Topics

- Enabling the As Run log
- Disabling the As Run log

Enabling the As Run log

To enable the As Run log, specify the channel name and enable the *As Run* log type for that channel.

Console

To enable the As Run log when creating a channel

1. Sign in to the AWS Management Console and open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.
3. On the navigation bar, choose **Create channel**.
4. In the **Set channel details**, **Configure outputs**, and **Access control** panes, configure your channel as desired.
5. In the **Access control** pane, choose **Next**.
6. In the **Logging** pane, under **Log types**, select **Enable as run** to enable the As Run log.

To enable the As Run log when updating a channel

> **Note**
>
> If the channel is currently running, you must first stop that channel before you can update it. After you stop the channel, you can choose **Actions** > **Edit** to begin updating the channel.

1. Sign in to the AWS Management Console and open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.
3. Choose the channel that you want to update to enable the As Run log for.
4. Choose **Actions** > **Edit**.
5. In the **Set channel details**, **Configure outputs**, and **Access control** panes, update your channel configuration as desired.
6. In the **Access control** pane, choose **Next**.
7. In the **Logging** pane, under **Log types**, select **Enable as run** to enable the As Run log.

To enable the As Run log from the Logging tab

> **Note**
>
> If the channel is currently running, you must use the **Logging** tab instead of choosing **Actions** > **Edit** to enable the As Run log.

1. Sign in to the AWS Management Console and open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.
3. Choose the channel that you want to enable the As Run log for.
4. In the navigation bar under the channel's name, choose **Logging**.
5. Under **Logging** > **Log types**, select **As run** to enable the As Run log.

AWS Command Line Interface (AWS CLI)

To enable the As Run log

Run the configure-logs-for-channel command and specify the appropriate values for the required parameters.

This example is formatted for Linux, macOS, or Unix, and it uses the backslash (\) line-continuation character to improve readability.

```
$ aws mediatailor configure-logs-for-channel \
--channel-name MyChannel \
--log-types AS_RUN
```

This example is formatted for Microsoft Windows, and it uses the caret (^) line-continuation character to improve readability.

```
C:\> aws mediatailor configure-logs-for-channel ^
--channel-name MyChannel ^
--log-types AS_RUN
```

Where:

- *MyChannel* is the name of the channel that you own and want to enable the As Run log for.

If the command runs successfully, you receive output similar to the following.

```
{
    "ChannelName": "MyChannel",
    "LogTypes": [
       "AS_RUN"
    ]
}
```

Disabling the As Run log

To disable the As Run log for a channel that has it enabled, specify the channel name and disable the *As Run* log type for that channel.

Console

To disable the As Run log when updating a channel

> **Note**
>
> If the channel is currently running, you must first stop that channel before you can update it. After you stop the channel, you can choose **Actions** > **Edit** to begin updating the channel.

1. Sign in to the AWS Management Console and open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.
3. Choose the channel that you want to update to enable the As Run log for.
4. Choose **Actions** > **Edit**.
5. In the **Set channel details**, **Configure outputs**, and **Access control** panes, update your channel configuration as desired.
6. In the **Access control** pane, choose **Next**.
7. In the **Logging** pane, under **Log types**, clear **Enable as run** to disable the As Run log.

To disable the As Run log from the Logging tab

> **Note**
>
> If the channel is currently running, you must use the **Logging** tab instead of choosing **Actions** > **Edit** to disable the As Run log.

1. Sign in to the AWS Management Console and open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. In the navigation pane, choose **Channel assembly** > **Channels**.
3. Choose the channel that you want to disable the As Run log for.
4. In the navigation bar under the channel's name, choose **Logging**.
5. Under **Logging** > **Log types**, clear **As run** to disable the As Run log.

AWS Command Line Interface (AWS CLI)

To disable the As Run log

Run the configure-logs-for-channel command and specify the appropriate values for the required parameters.

This example is formatted for Linux, macOS, or Unix, and it uses the backslash (\) line-continuation character to improve readability.

```
$ aws mediatailor configure-logs-for-channel \
--channel-name MyChannel \
--log-types
```

This example is formatted for Microsoft Windows, and it uses the caret (^) line-continuation character to improve readability.

```
C:\> aws mediatailor configure-logs-for-channel ^
--channel-name MyChannel ^
--log-types
```

Where:

- *MyChannel* is the name of the channel that you own and want to disable the As Run log for.

If the command runs successfully, you receive output similar to the following.

```
{
    "ChannelName": "MyChannel",
    "LogTypes": []
}
```

AWS Elemental MediaTailor ADS log analysis in Amazon CloudWatch Logs Insights

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

For general information, see Analyze log data with CloudWatch Logs insights.

> **Note**
>
> To access the logs, you need permissions to access Amazon CloudWatch. For instructions, see Permissions for Amazon CloudWatch Logs.

To view and query ADS logs using the CloudWatch console

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

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

Topics

- Querying the ADS logs

Querying the ADS logs

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

The following query retrieves all information from the ADS logs.

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

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

```
fields @timestamp, adsRequestUrl, requestHeaders.0.value as @userAgent,
 requestHeaders.1.value as @xForwardedFor, sessionId, requestId
| filter eventType = "MAKING_ADS_REQUEST"
| sort @timestamp asc
```

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

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

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

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

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

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

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

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

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

```
fields eventType
```

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

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

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

Controlling the volume of AWS Elemental MediaTailor logs

MediaTailor ad insertion session logs are sometimes verbose. To reduce log costs, you can define the percentage of session logs that MediaTailor sends to Amazon CloudWatch Logs. For example, if your playback configuration has 1000 ad insertion sessions and you set a percentage enabled value of 60, MediaTailor sends logs for 600 of the sessions to CloudWatch Logs. MediaTailor decides at random which of the sessions to send logs for. If you want to view logs for a specific session, you can use the debug log mode.

When you set a logging percentage, MediaTailor automatically creates a service-linked role that grants MediaTailor the permissions it requires to write CloudWatch Logs to your account. For information about how MediaTailor uses service-linked roles, see Using service-linked roles for MediaTailor.

Creating a log configuration

To control the percentage of session logs that MediaTailor writes to CloudWatch Logs, you create a *log configuration* for your playback configuration. When you create a log configuration, you specify a *playback configuration name*, and a *percent enabled* value.

Console

To create a log configuration for an *existing* playback configuration

1. Sign in to the AWS Management Console and open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. On the **Playback configuration** pane, select the playback configuration that you'd like to set the log configuration for.
3. Choose **Edit**.
4. Under **Log configuration**, specify a **percent enabled** value.

To create a log configuration for a *new* playback configuration

- Follow the procedure in Log configuration.

AWS Command Line Interface (AWS CLI)

To create a log configuration for an *existing* playback configuration

To create a log configuration by using the AWS CLI, run the configure-logs-for-playback-configuration command and specify the appropriate values for the required parameters.

This example is formatted for Linux, macOS, or Unix, and it uses the backslash (\) line-continuation character to improve readability.

```
$ aws mediatailor configure-logs-for-playback-configuration \
--percent-enabled 10 \
--playback-configuration-name MyPlaybackConfiguration
```

This example is formatted for Microsoft Windows, and it uses the caret (^) line-continuation character to improve readability.

```
C:\> aws mediatailor configure-logs-for-playback-configuration ^
--percent-enabled 10 ^
--playback-configuration-name MyPlaybackConfiguration
```

Where:

- *percent-enabled* is the percentage of playback configuration session logs that MediaTailor sends to CloudWatch Logs.
- *playback-configuration-name* is the name of the playback configuration to set the log configuration settings for.

If the command runs successfully, you receive output similar to the following.

```
{
    "PercentEnabled": 10,
    "PlaybackConfigurationName": "MyPlaybackConfiguration"
}
```

To create a log configuration for a *new* playback configuration

- Use the `configure-logs-for-playback-configuration` option for the put-playback-configuration command.

Deactivating a log configuration

After you create a log configuration, you can't delete it—you can only *deactivate* it. To deactivate the log configuration, set the **percent enabled** value to **0** with the MediaTailor console or API. This turns off all session logging for that playback configuration.

If you want to delete the service-linked role that MediaTailor uses for the log configuration(s) in your account, you must first deactivate all of your log configurations. For information about how to delete the service-linked role, see Using service-linked roles for MediaTailor.

Console

To deactivate log configuration on a playback configuration

1. Sign in to the AWS Management Console and open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. On the **Playback configuration** pane, select the playback configuration that you'd like to deactivate log configuration on.
3. Choose **Edit**.
4. Under **Log configuration**, set the **percent enabled** value to `0`. This turns off all session logging for this playback configuration.
5. Select **Save**.

AWS Command Line Interface (AWS CLI)

To deactivate a log configuration

- Set the `percent-enabled` value to `0` using the configure-logs-for-playback-configuration command.

Filtering AWS Elemental MediaTailor per-session logs and events

Logs emitted from a playback configuration in MediaTailor include information about a variety of activities that happen during the playback session. These activities are identified in the event type of the logs, and many events are logged by default. Because the volume of logs might impact your costs in Amazon CloudWatch, and you might want a different level of detail between sessions, MediaTailor provides controls over which event types are logged. With query parameters at session initialization, you can do the following:

- Specify the log events that you want to opt in to
- Enable logging raw responses from the ad decision server (ADS)

To define a customized level of log detail for each session, append the following parameters to your initial server-side or client-side playback session request. Add values to the parameters to represent the events that you want to include or exclude, in a comma-delimited format:

- `aws.adsInteractionLogPublishOptInEventTypes` to receive logs for specific ad decision server (ADS) interactions.
- `aws.adsInteractionLogExcludeEventTypes` to stop receiving logs for specific ADS interactions.
- `aws.manifestServiceLogExcludeEventTypes` to stop receiving logs for specific manifest service interactions.

> **Note**
>
> You must have debug mode enabled to receive any ManifestService logs. For information about debug log mode, including how to enable it, see Generating debug logs.

For a list of log and event types that MediaTailor emits, see Manifest logs.

If you don't pass through any query parameters for log filtering, MediaTailor writes all logs to your delivery destination.

Example server-side session initialization with log filters

To exclude GENERATED_MANIFEST and PARSING_ERROR events from your manifest logs and MAKING_ADS_REQUEST from the ADS logs, the session initialization request would look like this:

```
GET <mediatailorURL>/v1/master/<hashed-account-id>/<origin-id>/index.m3u8?
aws.logMode=DEBUG&aws.manifestServiceLogExcludeEventTypes=GENERATED_MANIFEST,PARSING_ERROR&aws.
```

To enable raw logs from your ADS, include the RAW_ADS_RESPONSE value for the AdsInteractionPublishOptInEventType parameter:

```
GET <mediatailorURL>/v1/master/<hashed-account-id>/<origin-id>/index.m3u8?
aws.adsInteractionPublishOptInEventType=RAW_ADS_RESPONSE
```

Example client-side session initialization with log filters

To exclude log events during client-side session initialization, include availSuppression and log-type parameters in your client's POST request to MediaTailor. For more information about how to construct a client-side playback session request, see Client-side ad tracking. The following example excludes CONFIG_SECURITY_ERROR and PARSING_ERROR events from your manifest logs and MAKING_ADS_REQUEST from the ADS logs.

```
POST parent.m3u8
   {
       "adsInteractionLog": {
           ...
           "excludeEventTypes": [
               "MAKING_ADS_REQUEST"
           ]
       },
       "manifestServiceLog": {
           ...
           "excludeEventTypes": [
               "GENERATED_MANIFEST",
               "PARSING_ERROR"
           ]
       },
      "logMode": "DEBUG"
   }
```

To enable raw logs from your ADS, include the `RAW_ADS_RESPONSE` value for the `publishOptInEventTypes` parameter:

```
POST parent.m3u8
   {
       "adsInteractionLog": {
           "publishOptInEventTypes": ["RAW_ADS_RESPONSE"],
           "excludeEventTypes": [
               "MAKING_ADS_REQUEST"
           ]
       },
       "manifestServiceLog": {
           ...
           "excludeEventTypes": [
               "GENERATED_MANIFEST",
               "PARSING_ERROR"
           ]
       },
       "logMode": "DEBUG"
   }
```

Generating AWS Elemental MediaTailor debug logs

Use debug logs to troubleshoot MediaTailor ad insertion playback session issues. To generate debug logs, set the log mode to debug in the player's request to MediaTailor. For server-side reporting, set the log mode in the *playback request*. For client-side reporting, set the log mode in the *session initialization request*.

When the log mode is set to debug, MediaTailor writes all log event types to CloudWatch Logs. The logs provide information about the following events. For a complete list of the data produced in the debug logs, see Debug log fields.

- **Origin interaction** – Details about MediaTailor interactions with the origin server. For example, the origin manifest response, manifest type, and origin URL.
- **Generated manifest** – Details about the playback session response from MediaTailor. For example, the manifest that MediaTailor generates.
- **Session initialized** – Session initialization details, such as the session ID.

To customize the log event types that you receive on a per-session basis, see Filtering logs and events.

Prerequisites

To set the log mode to debug, first you need to grant MediaTailor permission to send logs to CloudWatch, if you haven't already. Once you've granted permission for MediaTailor to access CloudWatch, then you're ready to enable the debug log mode. For information about how to grant MediaTailor permission to access CloudWatch see Setting Up Permissions for Amazon CloudWatch.

How to set the log mode to debug

This section explains how to set the log mode to debug for server-side reporting and client-side reporting.

Server-side reporting

For server-side reporting, include the `?aws.logMode=DEBUG` query parameter and value in your player's `GET` HTTP playback request to the HLS or DASH MediaTailor endpoint. For general information about server-side reporting see Server-side Reporting.

> **Important**
>
> The `DEBUG` value is case-sensitive.

A playback request that includes `?aws.logMode=DEBUG` looks like the following:

Example Playback request to an HLS endpoint

```
GET <mediatailorURL>/v1/master/<hashed-account-id>/<origin-id>/<asset-id>?
aws.logMode=DEBUG
```

After you set the log mode to debug, we recommend that you verify that the debug logging session is active. To verify that the debug session is active, check to see if there are any CloudWatch logs for the session ID. The session ID is included in the playback endpoint that MediaTailor provides. For more information, see Verify that the debug log mode is active for your playback session.

Client-side reporting

For client-side reporting, include the `logMode` key and `DEBUG` value in your client's `POST` HTTP session initialization request body to the MediaTailor /v1/session endpoint. For general information about client-side reporting see Client-Side Reporting.

Important

The DEBUG value is case-sensitive.

After you set the log mode to debug, we recommend that you verify that the debug session is active. To verify that the debug session is active, confirm that there's a `SESSION_INITIALIZED` event associated with the session ID in the CloudWatch logs. The session ID is included in the playback endpoint that MediaTailor provides. For more information, see Verify that the debug log mode is active for your playback session.

Maximum active debug sessions

You can have a maximum of 10 active debug log sessions. When your player sends its session initialization or playback request to MediaTailor, MediaTailor checks to see if the limit has been reached. If it has, MediaTailor checks to see if there are any stale sessions. A session is stale if it hasn't been accessed within a certain period of time. For live streams this time period is 10 minutes, for VOD streams it's 30 minutes.

If the maximum active debug log sessions limit has been reached, debug logs aren't written to CloudWatch Logs for your session. If you don't see debug logs in CloudWatch Logs for your session, you could have reached this limit. To confirm if the limit has been reached, see Verify that the debug log mode is active for your playback session.

Debug log fields

The following table lists the debug log fields that MediaTailor writes to CloudWatch.

Field	Description
`awsAccountId`	Your AWS account ID.
`customerId`	Your MediaTailor customer ID.
`eventTimestamp`	The ISO 8601 timestamp associated with the debug log event.
`eventType`	The type of debug log event. Values:

Field	Description
	• ORIGIN_INTERACTION – Details about MediaTailor interactions with the origin server. For example, the origin manifest response, manifest type, and origin URL. • GENERATED_MANIFEST – Details about the playback session response from MediaTailor. For example, the manifest that MediaTailor generates. • SESSION_INITIALIZED – Session initialization details, such as the session ID.
originRequestUrl	The URL of your origin server that is retrieved for this request.
mediaTailorPath	The MediaTailor endpoint that was called, including any parameters passed to MediaTailor in the initial manifest request.
requestId	The ID of a specific HTTP request to MediaTailor.
responseBody	The manifest in the response body from MediaTailor. This is either the raw origin manifest or the manifest generated by MediaTailor.
sessionId	The playback session ID.
sessionType	The type of playback session. Values: HLS, DASH

Read the debug logs

MediaTailor writes the debug logs to Amazon CloudWatch Logs. Typical CloudWatch Logs charges apply. Use CloudWatch Insights to read the debug logs. For information on how to use CloudWatch Logs Insights, see Analyzing Log Data with CloudWatch Logs Insights in the *AWS CloudWatch Logs User Guide*.

> **Note**
>
> The debug logs can take a few minutes to appear in CloudWatch. If you don't see the logs, wait a few minutes and try again. If you still don't see the logs, it could be that you've reached the maximum number of active debug log sessions. To verify if this is the case,

run a CloudWatch query to see if there was a debug session initialized for your playback session. For more information, see Verify that the debug log mode is active for your playback session.

Examples

This section includes example queries that you can use to read MediaTailor debug log data.

Example 1: Verify that the debug log mode is active for your playback session

```
fields @timestamp, @message
 | filter sessionId = "32002de2-837c-4e3e-9660-f3075e8dfd90"
 | filter eventType = "SESSION_INITIALIZED" # client-side reporting
 or mediaTailorPath like “/v1/master" # server-side reporting HLS
 or mediaTailorPath like “/v1/dash" # server-side reporting DASH
```

Example 2: View the responses from your origin

```
fields @timestamp, responseBody, @message, mediaTailorPath
 | filter eventType = "ORIGIN_MANIFEST" and sessionId = "32002de2-837c-4e3e-9660-
f3075e8dfd90"
```

Example 3: View the manifest generated by MediaTailor for a given session

```
fields @timestamp, responseBody, @message
 | filter mediaTailorPath like "/v1/master/" and eventType = "GENERATED_MANIFEST" and
 sessionId = "32002de2-837c-4e3e-9660-f3075e8dfd90"
```

Example 4: View all events for a given `requestId`

Use this query to view the origin manifest and the manifest generated by MediaTailor.

```
fields @timestamp, responseBody, @message, mediaTailorPath
 | filter requestId = "e5ba82a5-f8ac-4efb-88a0-55bed21c45b4"
```

Monitoring AWS Elemental MediaTailor with Amazon CloudWatch metrics

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

Metrics can be useful when you investigate stale manifests. For more information, see Using metrics to diagnose stale manifests.

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

To view metrics using the CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Under **All metrics**, choose the **MediaTailor** namespace.
4. Select the metric dimension to view the metrics (for example, **originID**).
5. Specify the time period that you want to view.

To view metrics using the AWS Command Line Interface (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.

Channel Assembly (CA) metrics

In the following table, all metrics are available by channel or by channel output.

Metric	Description
4xxErrorCount	The number of 4xx errors.
5xxErrorCount	The number of 5xx errors.
RequestCount	The total number of requests. The transaction count depends largely on how often players request updated manifests, and the number of players. Each player request counts as a transaction.
TotalTime	The amount of time that the application server took to process the request, including the time used to receive bytes from and write bytes to the client and network.

Server-side Ad-insertion (SSAI) metrics

The following table lists server-side ad-insertion metrics.

Metric	Description
AdDecisionServer.Ads	The count of ads included in ad decision server (ADS) responses within the CloudWatch time period that you specified.
AdDecisionServer.Duration	The total duration, in milliseconds, of all ads that MediaTailor received from the ADS within the CloudWatch time period that you specified. This duration can be greater than the Avail.Duration that you specified.

Metric	Description
AdDecisionServer.Errors	The number of non-HTTP 200 status code responses, empty responses, and timed-out responses that MediaTailor received from the ADS within the CloudWatch time period that you specified.
AdDecisionServer.FillRate	The simple average of the rates at which the responses from the ADS filled the correspon ding individual ad avails for the time period that you specified. To get the weighted average, calculate the AdDecisionServer.Duration as a percentage of the Avail.Duration . For more information about simple and weighted averages, see Simple and weighted averages.
AdDecisionServer.Latency	The response time in milliseconds for requests made by MediaTailor to the ADS.
AdDecisionServer.Timeouts	The number of timed-out requests to the ADS in the CloudWatch time period that you specified.
AdNotReady	The number of times that the ADS pointed at an ad that wasn't yet transcoded by the internal transcoder service in the time period that you specified. A high value for this metric might contribute to a low overall Avail.FillRate .
AdsBilled	The number of ads for which MediaTailor bills customers based on insertion.

Metric	Description
Avail.Duration	The planned total number of milliseconds of ad avails within the CloudWatch time period. The planned total is based on the ad avail durations in the origin manifest.
Avail.FilledDuration	The planned number of milliseconds of ad avail time that MediaTailor will fill with ads within the CloudWatch time period.
Avail.FillRate	The planned simple average of the rates at which MediaTailor will fill individual ad avails within the CloudWatch time period. To get the weighted average, calculate the Avail.FilledDuration as a percentage of the Avail.Duration . For more informati on about simple and weighted averages, see Simple and weighted averages. The maximum Avail.FillRate that MediaTailor can attain is bounded by the AdDecisionServer.FillRate . If the Avail.FillRate is low, compare it to the AdDecisionServer.FillRate . If the AdDecisionServer.FillRate is low, your ADS might not be returning enough ads for the avail durations.
Avail.Impression	The number of ads with impression tracking events that MediaTailor sees during server-si de beaconing (not the number of impressions).

Metric	Description
`Avail.ObservedDuration`	The observed total number of milliseconds of ad avails that occurred within the CloudWatch time period. `Avail.ObservedDuration` is emitted at the end of the ad avail, and is based on the duration of the segments reported in the manifest during the ad avail.
`Avail.ObservedFilledDuration`	The observed number of milliseconds of ad avail time that MediaTailor filled with ads within the CloudWatch time period.
`Avail.ObservedFillRate`	The observed simple average of the rates at which MediaTailor filled individual ad avails within the CloudWatch time period. Emitted only for HLS manifests, at the first `CUE-IN` tag. If there is no `CUE-IN` tag, MediaTailor doesn't emit this metric.
`Avail.ObservedSlateDuration`	The observed total number of milliseconds of slate that was inserted within the CloudWatch period.
`GetManifest.Age`	The total age of the manifest in milliseconds. Measured from when the origin creates the manifest, to when MediaTailor sends the personalized manifest. For more information about metrics for measuring manifest age, see Using metrics to diagnose stale manifests.
`GetManifest.Errors`	The number of errors received while MediaTailor was generating manifests in the CloudWatch time period that you specified.

Metric	Description
`GetManifest.Latency`	The MediaTailor response time in milliseconds for the request to generate manifests. For more information about metrics for measuring manifest age, see Using metrics to diagnose stale manifests.
`GetManifest.MediaTailorAge`	The amount of time that the manifest has been stored in MediaTailor in milliseconds. Measured from when MediaTailor receives an origin response, to when MediaTailor sends the personalized manifest. For more information about metrics for measuring manifest age, see Using metrics to diagnose stale manifests.
`Origin.Age`	The amount of time that the origin has the manifest in milliseconds. Measured from when the origin creates the manifest, to when MediaTailor sends the origin request. All `origin.*` metrics are emitted for requests that are fulfilled directly from the origin. They are not emitted for cached origin responses. For more information about metrics for measuring manifest age, see Using metrics to diagnose stale manifests.

Metric	Description
`Origin.Errors`	The number of non-HTTP 200 status code responses and timed-out responses that MediaTailor received from the origin server in the CloudWatch time period that you specified . All `origin.*` metrics are emitted for requests that are fulfilled directly from the origin. They are not emitted for cached origin responses.
`Origin.ManifestFileSizeBytes`	The file size of the origin manifest in bytes for both HLS and DASH. Typically this metric is used in conjunction with `Origin.ManifestFileSizeTooLarge` . All `origin.*` metrics are emitted for requests that are fulfilled directly from the origin. They are not emitted for cached origin responses.
`Origin.ManifestFileSizeTooLarge`	The number of responses from the origin that have a manifest size larger than the configured amount. Typically this metric is used in conjunction with `Origin.ManifestFileSizeBytes` . All `origin.*` metrics are emitted for requests that are fulfilled directly from the origin. They are not emitted for cached origin responses.
`Origin.Timeouts`	The number of timed-out requests to the origin server in the CloudWatch time period that you specified. All `origin.*` metrics are emitted for requests that are fulfilled directly from the origin. They are not emitted for cached origin responses.

Metric	Description
Requests	The number of concurrent transactions per second across all request types. The transacti on count depends mainly on the number of players and how often the players request updated manifests. Each player request counts as a transaction.
SkippedReason.DurationExceeded	The number of ads that were not inserted into an avail because the ADS returned a duration of ads that was greater than the specified avail duration. A high value for this metric might contribute to a discrepancy between the Avail.Ads and AdDecisionServer.A ds metric.
SkippedReason.EarlyCueIn	The number of ads skipped due to an early CUE-IN.
SkippedReason.ImportError	The number of ads skipped due to an error in the import job.
SkippedReason.ImportInProgress	The number of ads skipped due to an existing active import job.
SkippedReason.InternalError	The number of ads skipped due to a MediaTail or internal error.
SkippedReason.NewCreative	The number of ads that were not inserted into an avail because it was the first time the asset had been requested by a client. A high value for this metric might temporarily contribute to a low overall Avail.FillRate , until assets can be successfully transcoded.
SkippedReason.NoVariantMatch	The number of ads skipped due to there being no variant match between the ad and content.

Metric	Description
SkippedReason.PersonalizationThresholdExceeded	The duration of ads exceeding the **Personalization Threshold** setting in this configuration.
SkippedReason.ProfileNotFound	The number of ads skipped due to the transcoding profile not being found.
SkippedReason.TranscodeError	The number of ads skipped due to a transcode error.
SkippedReason.TranscodeInProgress	The count of the number of ads that were not inserted into an avail because the ad had not yet been transcoded. A high value for this metric might temporarily contribute to a low overall Avail.FillRate , until the assets can be successfully transcoded.

Simple and weighted averages

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

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

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

Example

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

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

The metrics are as follows:

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

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

AWS Elemental MediaTailor CloudWatch dimensions

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

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

Using metrics to diagnose stale manifests from AWS Elemental MediaTailor

A stale manifest is one that hasn't been recently updated. Different ad insertion workflows could have varying tolerance to how long must pass before a manifest is considered stale, based on a variety of factors (such as requirements for downstream systems). You can use Amazon CloudWatch metrics to identify manifests that exceed the staleness tolerance for your workflow, and help identify what could be causing the delays in manifest updates.

The following metrics help identify stale manifests and their causes.

For information about all metrics that MediaTailor emits, see AWS Elemental MediaTailor CloudWatch metrics.

Metric	Definition	Use
`GetManifest.Age`	Measures the total age of the manifest, including both `GetManifest.MediaTailorAge` and `Origin.Age` for this configuration.	You can use this metric to identify manifests that are past your update threshold and are stale. Set alarms on this metric so that you're alerted when stale manifests are being served. For information about alarms, see Alarming on metrics in the *Amazon CloudWatch User Guide.* When you receive an alarm, use `Origin.Age` and `GetManifest.MediaTailorAge` to identify if MediaTailor or the origin is causing the staleness.

Metric	Definition	Use
`Origin.Age`	Measures how long the origin has the manifest before sending it to MediaTailor for this configuration. This metric is not emitted when the response comes from a content delivery network (CDN). The response must come from the origin for `Origin.Age` to be emitted.	When you identify stale manifests with `GetManifest.Age`, you can analyze the `Origin.Age` metric and the `GetManifest.MediaTailorAge` metric to determine which is contributing to manifest staleness. If you find that `Origin.Age` is longer than your typical processing times at the origin, it's likely that the upstream system is causing the issue and you should focus diagnostics there.
`GetManifest.MediaTailorAge`	Measures how long MediaTailor has stored this manifest for this configuration.	When you identify stale manifests with `GetManifest.Age`, you can analyze the `GetManifest.MediaTailorAge` metric and the `Origin.Age` metric to determine which is contributing to manifest staleness. If the `GetManifest.MediaTailorAge` is longer than your typical manifest personalization times in MediaTailor, it's likely that MediaTailor is causing the issue and you should focus diagnostics there. `GetManifest.Latency` can further identify how long it takes for MediaTailor to create a personalized manifest.

Metric	Definition	Use
GetManifest.Latency	Measures the amount of time it takes for MediaTailor to process the request and create a personalized manifest for this configuration.	When you compare Origin.Age and GetManifest.MediaTailorAge and determine that MediaTailor is the cause of delayed manifest delivery, you can analyze the GetManifest.Latency metric to determine if the manifest personalization process is contributing to manifest staleness. GetManifest.MediaTailorAge measures the total time that the manifest is stored in MediaTailor. GetManifest.Latency measures how much of that storage time is MediaTailor personalizing the manifest in response to a request.

Recording AWS Elemental MediaTailor API calls

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

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

AWS Elemental MediaTailor information in CloudTrail

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

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

- Creating a trail for your AWS account
- AWS service integrations with CloudTrail logs
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

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

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

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

For more information, see CloudTrail userIdentity element.

Understanding AWS Elemental MediaTailor log file entries

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

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


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAEXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/testuser",
        "accountId": "111122223333",
        "accessKeyId": "AIDAEXAMPLE",
        "userName": "testuser"
    },
    "eventTime": "2018-12-28T22:53:46Z",
    "eventSource": "mediatailor.amazonaws.com",
    "eventName": "PutPlaybackConfiguration",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "1.2.3.4",
    "userAgent": "PostmanRuntime/7.4.0",
    "requestParameters": {
        "VideoContentSourceUrl": "http://examplevideo.com",
        "Name": "examplename",
        "AdDecisionServerUrl": "http://exampleads.com"
    },
    "responseElements": {
        "SessionInitializationEndpointPrefix": "https://777788889999.mediatailor.us-
east-1.amazonaws.com/v1/session/AKIAIOSFODNN7EXAMPLE/examplename/",
        "DashConfiguration": {
            "ManifestEndpointPrefix": "https://777788889999.mediatailor.us-
east-1.amazonaws.com/v1/dash/AKIAIOSFODNN7EXAMPLE/examplename/",
            "MpdLocation": "EMT_DEFAULT"
        },
        "AdDecisionServerUrl": "http://exampleads.com",
        "CdnConfiguration": {},
```

```
        "PlaybackEndpointPrefix": "https://777788889999.mediatailor.us-
east-1.amazonaws.com",
        "HlsConfiguration": {
            "ManifestEndpointPrefix": "https://777788889999.mediatailor.us-
east-1.amazonaws.com/v1/master/AKIAIOSFODNN7EXAMPLE/examplename/"
        },
        "VideoContentSourceUrl": "http://examplevideo.com",
        "Name": "examplename"
    },
    "requestID": "1a2b3c4d-1234-5678-1234-1a2b3c4d5e6f",
    "eventID": "987abc65-1a2b-3c4d-5d6e-987abc654def",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

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

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

```
    "recipientAccountId": "111122223333"
}
```

Receiving AWS Elemental MediaTailor Channel Assembly alerts

MediaTailor creates alerts for issues or potential issues that occur with your channel assembly resources. The alert describes the issue, when the issue occurred, and the affected resources.

You can view the alerts in the AWS Management Console, the AWS Command Line Interface (AWS CLI), AWS SDKs, or programmatically using the MediaTailor ListAlerts API.

> **Important**
>
> Alerts are only available for channel assembly resources created on or after July 14th, 2021.

Channel Assembly Alerts

Alert Type	Alert Code	Alert Message	Notes
VOD Source	NOT_PROCESSED	MediaTailor hasn't processed the package configuration *configurationPath* .	
	UNREACHABLE	We can't reach the URL *url*.	
	UNAUTHORIZED	*url* didn't authorize the request.	
	TIMEOUT	The connection to *url* timed out.	
	UNPARSABLE_MANIFEST	MediaTailor encountered an issue while parsing the manifest from *url*.	

Alert Type	Alert Code	Alert Message	Notes
	VARIANT_DURATION_MISMATCH	MediaTailor encountered variants with mismatched total durations while parsing the manifest from *url*. This might cause stalling during playback.	Your manifest has varying durations across variants/representations. This might result in missing or incorrect captions, and MediaTailor being unable to insert ads.
	SEGMENT_DURATION_TOO_LONG	MediaTailor encountered a segment with a duration greater than thirty seconds while parsing the manifest from *url*. This might cause stalling during playback, missing or incorrect captions, and the inability to insert advertisements.	Your manifest contains a segment that is greater than 30 seconds.
	TARGET_DURATION_MISMATCH	MediaTailor encountered a mismatch of EXT-X-TARGETDURATION values across HLS manifests while parsing the manifest from *url*. This might cause stalling during playback.	The target duration doesn't match across all manifests in the source..
Source Location	NOT_PROCESSED	MediaTailor hasn't processed the resource *resourceName* .	
Program	VOD_SOURCE_ALERT	The VOD source *vodSourceName* in this program has the following alert: *vodSourceAlertCode* : *vodSourceAlertMessage*	

Alert Type	Alert Code	Alert Message	Notes
	SOURCE_LOCATION_ALERT	The source location *sourceLocationName* contained in this program has the following alert: *sourceLocationAlertCode* : *sourceLocationAlertMessage*	
	CODEC_MISMATCH	MediaTailor encountered mismatched codec in *channelName* schedule. The mismatch is in *sourceGroupName* between *programName1* 's manifest *manifestUrl* and *programName2* 's manifest *manifestUrl* .	
	RESOLUTION_MISMATCH	MediaTailor encountered mismatched resolution in *channelName* schedule. The mismatch is in *sourceGroupName* between *programName1* 's manifest *manifestUrl* and *programName2* 's manifest *manifestUrl* .	

Alert Type	Alert Code	Alert Message	Notes
	BANDWIDTH_MISMATCH	MediaTailor encountered mismatched bandwidth in *channelName* schedule. The mismatch is in *sourceGroupName* between *programName1* 's manifest *manifestUrl* and *programName2* 's manifest *manifestUrl* .	
	FRAMERATE_MISMATCH	MediaTailor encountered mismatched framerate in *channelName* schedule. The mismatch is in *sourceGroupName* between *programName1* 's manifest *manifestUrl* and *programName2* 's manifest *manifestUrl* .	
	TARGET_DURATION_MISMATCH	MediaTailor encountered mismatched EXT-X-TARGETDURATION values across HLS manifests in *channelName* schedule. The mismatch is in *sourceGroupName* between *programName1* 's manifest *manifestUrl* and *programName2* 's manifest *manifestUrl* .	

Alert Type	Alert Code	Alert Message	Notes
	SEGMENT_DURATION_MISMATCH	MediaTailor encountered mismatched segment duration values across manifests in *channelName* schedule. The mismatch is in *sourceGroupName* between *programName1* 's manifest *manifestUrl* and *programName2* 's manifest *manifestUrl* .	
	NO_COMMON_SEGMENT_BOUNDARY_FOR_AD_SLATE	MediaTailor was not able to insert ad slate at offset *offsetMillis* for program *programName* . There is no common segment boundary at the ad slate's start time.	
	NOT_PROCESSED	MediaTailor hasn't processed the resource *resourceName* .	
	TOO_MANY_ALERTS	MediaTailor has found too many alerts and will not provide any more alerts for *programName* . Clear existing alerts to continue receiving alerts for *programName* .	
Channel	PROGRAM_ALERT	The program *programName* contained in this channel has the following alert: *programAlertCode* : *programAlertMessage*	

Viewing alerts

You can view alerts for any MediaTailor channel assembly resource. When you view the alerts for channels and programs, MediaTailor includes all of the related resources contained within the channel or program. For example, when you view the alerts for a specific program, you also see alerts for the source location and VOD sources that the program contains.

To view alerts, perform the following procedure.

Console

To view alerts in the console

1. Open the MediaTailor console at https://console.aws.amazon.com/mediatailor/.
2. Choose the resource that you want to view alerts for.
3. Select the **Alerts** tab to view the alerts.

AWS Command Line Interface (AWS CLI)

To list alerts for a channel assembly resource, you need the resource's Amazon Resource Name (ARN). You can use the `describe-`*`resource_type`* command in the AWS Command Line Interface (AWS CLI) to get the resource's ARN. For example, run the describe-channel command to get a specific channel's ARN:

```
aws mediatailor describe-channel --channel-name MyChannelName
```

Then use the aws mediatailor list-alerts command to list the alerts associated with the resource:

```
aws mediatailor list-alerts --resource-arn arn:aws:mediatailor:region:aws-account-
id:resource-type/resource-name
```

API

To list alerts for a channel assembly resource, you need the resource's Amazon Resource Name (ARN). You can use the `Describe`*`Resource`* operation in the MediaTailor API to get the resource's ARN. For example, use the DescribeChannel operation to get a specific channel's ARN.

Then use the ListAlerts API to list the alerts for the resource.

Handling alerts

When an alert occurs, view the alerts in the AWS Management Console, or use the AWS Command Line Interface (AWS CLI), AWS SDKs, or the MediaTailor Alerts API to determine the possible sources of the issue.

After you resolve the issue, MediaTailor clears the alert.

Tagging AWS Elemental MediaTailor resources

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

Tags help you do the following:

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

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

Supported resources in AWS Elemental MediaTailor

The following resources in AWS Elemental MediaTailor supports tagging:

- Channels
- Configurations
- SourceLocations
- VodSources

Tag restrictions

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

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

Managing tags in AWS Elemental MediaTailor

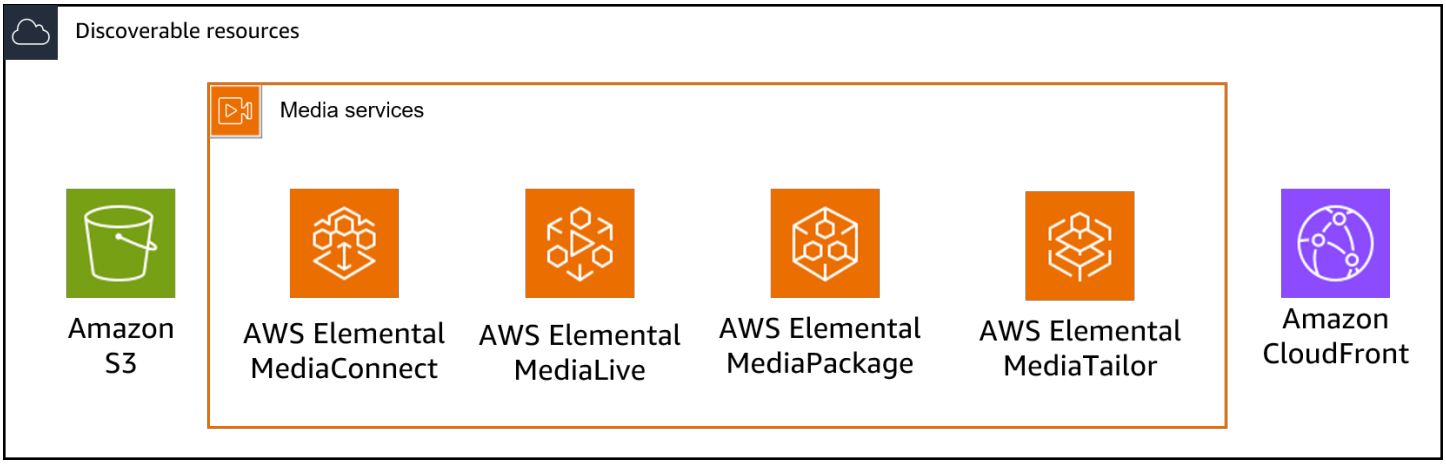
You set tags as properties on a resource. You can add, edit, and delete tags through the AWS Elemental MediaTailor API or the AWS Command Line Interface (AWS CLI). For more information, see the AWS Elemental MediaTailor API reference.

Monitoring AWS media services with workflow monitor

Workflow monitor is a tool for the discovery, visualization, and monitoring of AWS media workflows. Workflow monitor is available in the AWS console and API. You can use workflow monitor to discover and create visual mappings of your workflow's resources, called *signal maps*. You can create and manage Amazon CloudWatch alarm and Amazon EventBridge rule templates to monitor the mapped resources. The monitoring templates you create are transformed into deployable AWS CloudFormation templates to allow repeatability. AWS recommended alarm templates provide predefined best-practice monitoring.

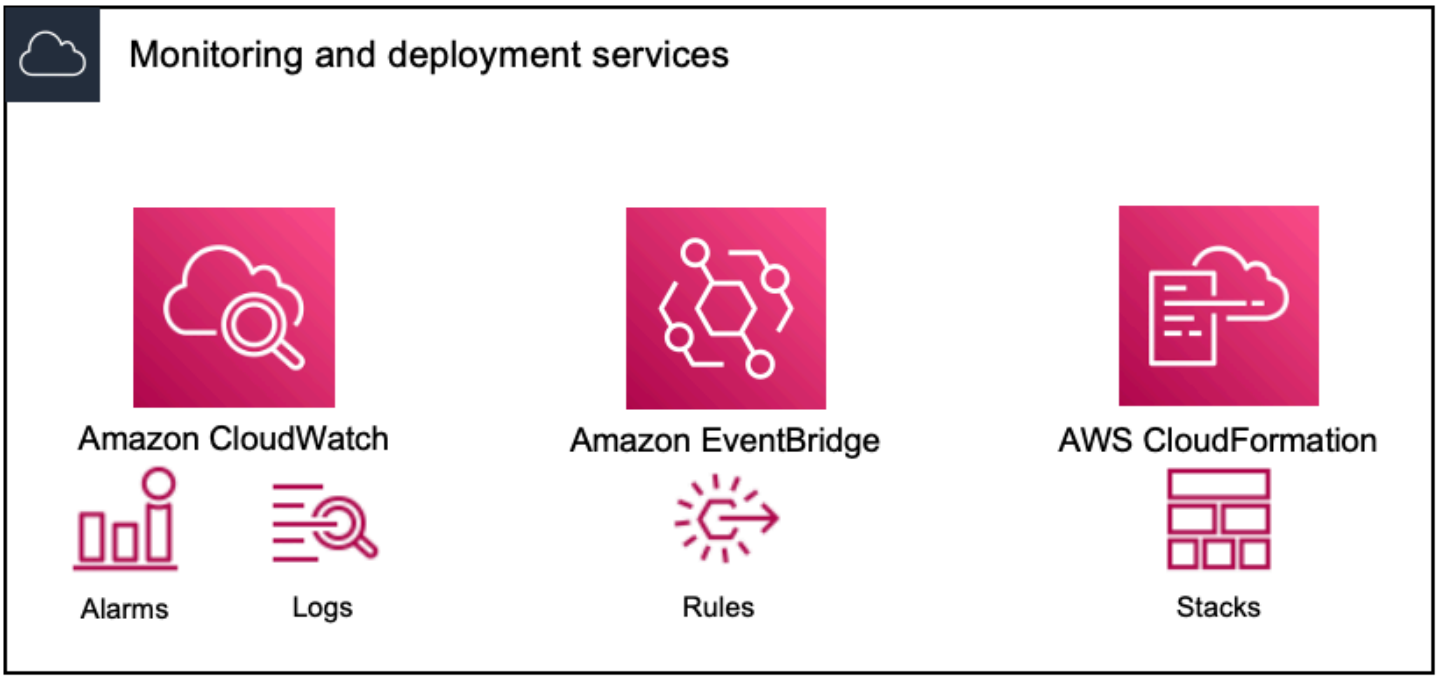
Discover

Utilize signal maps to automatically discover interconnected AWS resources associated with your media workflow. Discovery can begin at any supported service resource and creates an end-to-end mapping of the workflow. Signal maps can be used as stand-alone visualization tools or enhanced with monitoring templates.



Monitor

You can create custom CloudWatch alarm and EventBridge rule templates to monitor the health and status of your media workflows. Best practice alarm templates are available to import into your workflow monitor environment. You can use the best practice alarm templates as they are, or edit them to better fit your workflow. Any templates you create are transformed into AWS CloudFormation templates for repeatable deployment.



> **Note**
>
> There is no direct cost for using workflow monitor. However, there are costs associated with the resources created and used to monitor your workflow.

When monitoring is deployed, Amazon CloudWatch and Amazon EventBridge resources are created. When using the AWS Management Console, prior to deploying monitoring to a signal map, you will be notified of how many resources will be created. For more information about pricing, see: CloudWatch pricing and EventBridge pricing.
Workflow monitor uses AWS CloudFormation templates to deploy the CloudWatch and EventBridge resources. These templates are stored in a standard class Amazon Simple Storage Service bucket that is created on your behalf, by workflow monitor, during the deployment process and will incur object storage and recall charges. For more information about pricing, see: Amazon S3 pricing.
Previews generated in the workflow monitor signal map for AWS Elemental MediaPackage channels are delivered from the MediaPackage Origin Endpoint and will incur Data Transfer Out charges. For pricing, see: MediaPackage pricing.

Components of workflow monitor

Workflow monitor has four major components:

- CloudWatch alarm templates - Define the conditions you would like to monitor using CloudWatch. You can create your own alarm templates, or import predefined templates created by AWS. For more information, see: CloudWatch alarm groups and templates for monitoring your AWS media workflow
- EventBridge rule templates - Define how EventBridge sends notifications when an alarm is triggered. For more information, see: EventBridge rule groups and templates for monitoring your AWS media workflow
- Signal maps - Use an automated process to create AWS Elemental workflow maps using existing AWS resources. The signal maps can be used to discover resources in your workflow and deploy monitoring to those resources. For more information, see: Workflow monitor signal maps
- Overview - The overview page allows you to directly monitor the status of multiple signal maps from one location. Review metrics, logs, and alarms for your workflows. For more information, see: Workflow monitor overview

Supported services

Workflow monitor supports automatic discovery and signal mapping of resources associated with the following services:

- AWS Elemental MediaConnect
- AWS Elemental MediaLive
- AWS Elemental MediaPackage
- AWS Elemental MediaTailor
- Amazon S3
- Amazon CloudFront

Topics

- Configuring workflow monitor to monitor AWS media services
- Using workflow monitor

Configuring workflow monitor to monitor AWS media services

To setup workflow monitor for the first time; you create the alarm and event templates, and discover signal maps that are used to monitor your media workflows. The following guide contains the steps necessary to setup both Administrator and Operator level IAM roles, create workflow monitor resources, and deploy monitoring to your workflows.

Topics

- Getting started with workflow monitor
- Workflow monitor groups and templates
- Workflow monitor signal maps
- Workflow monitor quotas

Getting started with workflow monitor

The following steps provide a basic overview of using workflow monitor for the first time.

1. Setup workflow monitor IAM permissions for administrator and operator level roles: Workflow monitor IAM policies
2. Build alarm templates or import predefined templates created by AWS: CloudWatch alarms
3. Build notification events that will be delivered by EventBridge: EventBridge rules
4. Discover signal maps using your existing AWS Elemental resources: Signal maps
5. Attach the alarm templates and notification rules to your signal map: Attaching templates

6. Deploy the templates to begin monitoring the signal map: Deploying monitoring templates
7. Monitor and review your workflow monitor resources using the overview section of the AWS console: Overview

Workflow monitor IAM policies

Workflow monitor interacts with multiple AWS services to create signal maps, build CloudWatch and EventBridge resources, and AWS CloudFormation templates. Because workflow monitor interacts with a wide range of services, specific AWS Identity and Access Management (IAM) policies must be assigned for these services. The following examples indicate the necessary IAM policies for both administrator and operator IAM roles.

Administrator IAM policy

The following example policy is for an administrator-level workflow monitor IAM policy. This role allows for the creation and management of workflow monitor resources and the supported service resources that interact with workflow monitor.

```
                    {
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:List*",
      "cloudwatch:Describe*",
      "cloudwatch:Get*",
      "cloudwatch:PutAnomalyDetector",
      "cloudwatch:PutMetricData",
      "cloudwatch:PutMetricAlarm",
      "cloudwatch:PutCompositeAlarm",
      "cloudwatch:PutDashboard",
      "cloudwatch:DeleteAlarms",
      "cloudwatch:DeleteAnomalyDetector",
```

```
        "cloudwatch:DeleteDashboards",
        "cloudwatch:TagResource",
        "cloudwatch:UntagResource"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:List*",
        "cloudformation:Describe*",
        "cloudformation:CreateStack",
        "cloudformation:UpdateStack",
        "cloudformation:DeleteStack",
        "cloudformation:TagResource",
        "cloudformation:UntagResource"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudfront:List*",
        "cloudfront:Get*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeNetworkInterfaces"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:List*",
        "events:Describe*",
        "events:CreateEventBus",
        "events:PutRule",
        "events:PutTargets",
        "events:EnableRule",
        "events:DisableRule",
```

```
        "events:DeleteRule",
        "events:RemoveTargets",
        "events:TagResource",
        "events:UntagResource"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:Describe*",
        "logs:Get*",
        "logs:TagLogGroup",
        "logs:TagResource",
        "logs:UntagLogGroup",
        "logs:UntagResource"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "mediaconnect:List*",
        "mediaconnect:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "medialive:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "mediapackage:List*",
        "mediapackage:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "mediapackagev2:List*",
        "mediapackagev2:Get*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "mediapackage-vod:List*",
        "mediapackage-vod:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "mediatailor:List*",
        "mediatailor:Describe*",
        "mediatailor:Get*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "resource-groups:ListGroups",
        "resource-groups:GetGroup",
        "resource-groups:GetTags",
        "resource-groups:GetGroupQuery",
        "resource-groups:GetGroupConfiguration",
        "resource-groups:CreateGroup",
        "resource-groups:UngroupResources",
        "resource-groups:GroupResources",
        "resource-groups:DeleteGroup",
        "resource-groups:UpdateGroupQuery",
        "resource-groups:UpdateGroup",
        "resource-groups:Tag",
        "resource-groups:Untag"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "s3:*"
      ],
      "Resource": "arn:aws:s3:::workflow-monitor-templates*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:TagResource",
        "sns:UntagResource"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "tag:Get*",
        "tag:Describe*",
        "tag:TagResources",
        "tag:UntagResources"
      ],
      "Resource": "*"
    }
  ]
}
```

Operator IAM policy

The following example policy is for an operator-level workflow monitor IAM policy. This role allows for limited and read-only access to the workflow monitor resources and the supported service resources that interact with workflow monitor.

```
                  {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:List*",
```

```
        "cloudwatch:Describe*",
        "cloudwatch:Get*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:List*",
        "cloudformation:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudfront:List*",
        "cloudfront:Get*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeNetworkInterfaces"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:List*",
        "events:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:Describe*",
        "logs:Get*"
      ],
      "Resource": "*"
    },
```

```
    {
      "Effect": "Allow",
      "Action": [
        "mediaconnect:List*",
        "mediaconnect:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "medialive:List*",
        "medialive:Get*",
        "medialive:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "mediapackage:List*",
        "mediapackage:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "mediapackagev2:List*",
        "mediapackagev2:Get*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "mediapackage-vod:List*",
        "mediapackage-vod:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "mediatailor:List*",
        "mediatailor:Describe*",
        "mediatailor:Get*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "arn:aws:s3:::workflow-monitor-templates*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "tag:Get*",
        "tag:Describe*"
      ],
      "Resource": "*"
    }
  ]
}
```

Workflow monitor groups and templates

Before you can deploy workflow monitoring to a signal map, you must create the groups and templates for CloudWatch alarms and EventBridge notifications. The CloudWatch templates define what scenarios and thresholds will be used to trigger the alarms. The EventBridge templates will determine how these alarms are reported to you.

If you only want mappings of your connected resources and do not want to use the monitoring template capabilities of workflow monitor, signal maps can be used without CloudWatch and EventBridge templates. For more information about using signal maps, see: Signal maps

Topics

- CloudWatch alarm groups and templates for monitoring your AWS media workflow
- EventBridge rule groups and templates for monitoring your AWS media workflow

CloudWatch alarm groups and templates for monitoring your AWS media workflow

Workflow monitor alarms allow you to use existing CloudWatch metrics as the foundation of alarms for your signal maps. You can create an alarm template group to sort and classify the types of alarming that is important to your workflow. Within each alarm template group, you create alarm templates with specific CloudWatch metrics and parameters that you want to monitor. You can create your own alarm templates or import recommended alarm templates created by AWS. After creating an alarm template group and alarm templates within that group, you can attach one or more of these alarm template groups to a signal map.

You must create an alarm template group first. After you have created an alarm template group, you can create your own templates or use recommended templates created by AWS. If you want to create your own alarm templates, continue on this page. For more information about importing recommended templates, see: Recommended templates

This section covers the creation of CloudWatch alarms using workflow monitor. For more information about how the CloudWatch service handles alarms and details of the alarm components, see: Using CloudWatch alarms in the *Amazon CloudWatch User Guide*

Creating alarm template groups

Alarm template groups allow you to sort and classify the types of alarms that are important to your workflow.

To create an alarm template group

1. From the workflow monitor console's navigation pane, select **CloudWatch alarm templates**.
2. Select **Create alarm template group**.
3. Give the alarm template group a unique **Group name** and optional **Description**.
4. Select **Create**, You will be taken to the newly created alarm template group's details page.

Creating alarm templates

You can create alarm templates with the CloudWatch metrics and parameters that you want to monitor.

To create an alarm template

1. From the alarm template group's details page, select **Create alarm template**.

2. Give the alarm template a unique **Template name** and optional **Description**.
3. In the **Choose metric** section:
    1. Select a **Target Resource Type**. The target resource type is a resource for the respective service, such as a channel for MediaLive and MediaPackage or a flow for MediaConnect.
    2. Select a **Metric Name**. This is the CloudWatch metric that acts as the foundation for the alarm. The list of metrics will change depending on the selected **Target Resource Type**.
4. In the **Alarm settings** section:

    > **Note**
    >
    > For more information about how the CloudWatch service handles alarms and details of the alarm components, see: Using CloudWatch alarms in the *Amazon CloudWatch User Guide*

    1. Select the **Statistic**. This is a value such as a **Sum** or an **Average** that will be used to monitor the metric.
    2. Select the **Comparison Operator**. This field references the **Threshold** that you set in the next step.
    3. Set a **Threshold**. This is a numeric value that the **Comparison Operator** uses to determine greater than, less than, or equal to status.
    4. Set a **Period**. This is a time value, in seconds. The **Period** is the length of time that the **Statistic**, **Comparison Operator**, and **Threshold** interact to determine if the alarm gets triggered.
    5. Set the **Datapoints**. This value determines how many datapoints are needed to trigger the alarm.
    6. Select how to **Treat Missing Data**. This selection determines how this alarm reacts to missing data.
5. Select **Create** to complete the process.

An example of a completed alarm template could have the following parameters: A MediaConnect flow **Target Resource Type** is monitored for the Disconnections **Metric Name**. The **Statistic** value is set to Sum with a **Comparison Operator** of "greater than or equal to" and a **Threshold** of 10. The

Period is set to 60 seconds, and only requires 1 out of 1 **Datapoints**. **Treat Missing Data** is set to "ignore."

The result of these settings is: workflow monitor will monitor for disconnections on the flow. If 10 or more disconnections occur within 60 seconds, the alarm will be triggered. 10 or more disconnections in 60 seconds only needs to happen one time for the alarm to be triggered.

Recommended alarm templates for monitoring your AWS media workflow

Workflow monitor's recommended templates are a curated selection of AWS Elemental service metrics with predefined alarm settings appropriate for the metric. If you do not want to create customized alarm templates, recommended templates provide you with best-practice monitoring templates that are created by AWS.

Workflow monitor contains recommended template groups for each supported service. These groups are designed to apply best-practice monitoring to specific types of workflows. Each template group contains a curated selection of alarms configured from service-specific metrics. For example, a recommended template group for a MediaLive multiplex workflow will have a different set of preconfigured metrics than a MediaConnect CDI workflow.

To use recommended alarm templates

1. Follow the steps to create an alarm template group, or select an existing one.
2. In the **Alarm templates** section, select **Import**. You will need to import the AWS recommended templates into your template group.
3. Use the **CloudWatch alarm template groups** dropdown to select an AWS recommended group. These groups contain curated alarms for specific services.
4. Select the templates to import using the check boxes. Each template will list its metrics, preconfigured monitoring values, and provide a description of the metric. When you are done selecting templates, select the **Add** button.
5. The selected templates will move to the **Alarm template(s) to import** section. Review your choices and select **Import**.
6. After the import is complete, the selected templates will be added to the template group. If you want to add more templates, repeat the import process.
7. Imported templates can be customized after import. Alarm settings can be modified to fit your alarming needs.

EventBridge rule groups and templates for monitoring your AWS media workflow

CloudWatch uses Amazon EventBridge rules to send notifications. You begin by creating an event template group. In that event template group, you create event templates that determine what conditions create a notification and who is notified.

This section covers the creation of EventBridge rules using workflow monitor. For more information about how the EventBridge service uses rules, see: EventBridge rules in the *Amazon EventBridge User Guide*

Creating event template groups

Event template groups allow you to sort and classify events based on your use case.

To create an event template group

1. From the workflow monitor console's navigation pane, select **EventBridge rule templates**.
2. Select **Create event template group**.
3. Give the alarm template group a unique **Group name** and optional **Description**.
4. Select **Create**, You will be taken to the newly created alarm template group's details page.

Creating event templates

You can send notifications based on event templates you create.

To create an event template

1. From the event template group's details page, select **Create event template**.
2. Give the event template a unique **Template name** and optional **Description**.
3. In the **Rule settings** section:
    1. Select an **Event type**. When selecting an event type, you can choose between several events created by AWS or select **Signal map active alarm** to use an alarm created by an alarm template.
    2. Select a **Target service**. This determines how you would like to be notified of this event. You can select Amazon Simple Notification Service or CloudWatch logs.
    3. After selecting a target service, select a **Target**. This will be a Amazon SNS topic or a CloudWatch log group, depending on your target service selection.

4. Select **Create** to complete the process.

Workflow monitor signal maps

Signal maps are visual mappings of AWS resources in your media workflow. You can use workflow monitor to start the signal map discovery on any of the supported resource types. During the discovery process, workflow monitor will automatically and recursively map all connected AWS resources. After the signal map has been created, you can use the workflow monitor console to do things like deploy monitoring templates, view metrics, and view details of the mapped resources.

Topics

- Creating signal maps for AWS media workflows
- Viewing signal maps of AWS media workflows
- Attaching alarm and event templates to the signal map of your AWS media workflow
- Deploying templates to the signal map of your AWS media workflow
- Updating the signal map of your AWS media workflow
- Deleting the signal map of your AWS media workflow

Creating signal maps for AWS media workflows

You can use workflow monitor signal maps to create a visual mapping of all connected AWS resources in your media workflow.

To create a signal map

1. From the workflow monitor console's navigation pane, select **Signal maps**.
2. Select **Create signal map**.
3. Give the signal map a **Name** and **Description**.
4. In the **Discover new signal map** section, resources in the current account and selected region are displayed. Select a resource to begin signal map discovery. The selected resource will be the starting point for discovery.
5. Select **Create**. Allow a few moments for the discovery process to complete. After the process is complete, you will be presented with the new signal map.

> **Note**
>
> Previews generated in the workflow monitor signal map for AWS Elemental MediaPackage channels are delivered from the MediaPackage Origin Endpoint and will incur Data Transfer Out charges. For pricing, see: MediaPackage pricing.

Viewing signal maps of AWS media workflows

Workflow monitor signal maps allow you to see a visual mapping of all connected AWS resources in your media workflow.

Signal map views

After selecting a signal map, you have two views that can be used to monitor or configure the signal map. **Monitor signal map** and **Configure signal map** is a context-sensitive button found in the upper-right of the signal map console section.

If you select the signal map using the **Signal maps** section of the navigation pane, your signal map will be displayed in the configuration view. The configuration view allows you to make changes to the template groups attached to this signal map, deploy the attached templates, and view the basic details and tags of the signal map.

If you select the signal map using the **Overview** section of the navigation pane, your signal map will be displayed in monitoring view. The monitoring view displays the CloudWatch alarms, EventBridge rules, alerts, logs, and metrics for this signal map.

The view can be changed at any time by selecting the **Monitor/Configure signal map** button in the upper-right. The configuration view requires administrator-level IAM permissions. Required IAM permissions can be viewed here: Workflow monitor IAM policies

Navigating the signal map

A signal map will contain nodes for every supported AWS resource discovered by workflow monitor. Certain resources, such as MediaLive channels and MediaPackage endpoints can display thumbnail previews of the content, if thumbnail previews are available.

Selecting a resource node, and selecting **View selected resource details** from the **Actions** dropdown menu will take you to the associated service's details page. For example, selecting a

MediaLive channel and selecting **View selected resource details** will open the MediaLive console's details page for that channel.

Selecting a resource node will filter the list of active alarms to only that node. If you select the resource's **Target ARN** in the active alarm, you will be taken to the associated service's details page, with the selected resource open.

Attaching alarm and event templates to the signal map of your AWS media workflow

After you have created alarm and event templates, you need to attach these to a signal map. Any of the alarm and event templates you have created can be attached to any discovered signal maps.

To attach alarm and event templates to your signal map

1. From the workflow monitor console's navigation pane, select **Signal maps** and select the signal map you want to work with.
2. In the upper-right of the signal map page, in the **CloudWatch alarm template groups** tab, select **Attach CloudWatch alarm template groups**.
    1. In the new section that opens, choose all of the alarm template groups that you want to apply to this signal map, then select **Add**. This will cause the selected alarm template groups to move to the **Attached CloudWatch alarm template groups** section.
    2. Selecting **Save** will save your changes and return you to the signal map page.
3. At the right of the signal map page, select the **EventBridge rule template groups** tab then select **Attach EventBridge rule template groups**.
    1. In the new section that opens, choose all of the event template groups that you want to apply to this signal map, then select **Add**. This will cause the selected rule template groups to move to the **Attached EventBridge rule template groups** section.
    2. Selecting **Save** will save your changes and return you to the signal map page.
4. You have assigned CloudWatch alarm and EventBridge rule templates to the signal map, but the monitoring is not yet deployed. The next section will cover the deployment of the monitoring resources.

Deploying templates to the signal map of your AWS media workflow

After you have attached the alarm and event templates to your signal map, you must deploy the monitoring. Until the deployment is complete, the monitoring of your signal map will not be active.

Workflow monitor will only deploy alarms that are relevant to the selected signal map. For example, the attached alarm template group might contain alarms for multiple services, such as MediaLive, MediaPackage, and MediaConnect. If the selected signal map only contains MediaLive resources, no MediaPackage or MediaConnect alarms will be deployed.

To deploy the monitoring templates

1. After attaching alarm and event template groups to your signal map and saving your changes, select **Deploy monitor** in the **Actions** dropdown menu.
2. You will be asked to confirm the deployment and presented with the number of CloudWatch and EventBridge resources that will be created. If you would like to proceed, select **Deploy**.

   > **Note**
   >
   > There is no direct cost for using workflow monitor. However, there are costs associated with the resources created and used to monitor your workflow.
   > When monitoring is deployed, Amazon CloudWatch and Amazon EventBridge resources are created. When using the AWS Management Console, prior to deploying monitoring to a signal map, you will be notified of how many resources will be created. For more information about pricing, see: CloudWatch pricing and EventBridge pricing.
   > Workflow monitor uses AWS CloudFormation templates to deploy the CloudWatch and EventBridge resources. These templates are stored in a standard class Amazon Simple Storage Service bucket that is created on your behalf, by workflow monitor, during the deployment process and will incur object storage and recall charges. For more information about pricing, see: Amazon S3 pricing.

3. The status of the deployment is displayed next to the name of the signal map. The deployment status is also visible in the **Stacks** section of the AWS CloudFormation console. After a few moments of resource creation and deployment, your signal map monitoring will begin.

Updating the signal map of your AWS media workflow

If a change is made to your workflow, you might need to rediscover the signal map and redeploy monitoring resources. Workflow monitor is a visualization and monitoring tool that does not have the ability to make any changes to your workflow. Signal maps represent a point-in-time visualization of your workflow. In the event that you add, remove, or significantly modify parts of your media workflow, we recommend that you rediscover the signal map. If you have monitoring resources attached to the signal map, we recommend you redeploy monitoring after the rediscovery process.

To rediscover a signal map

1. From the workflow monitor console's navigation pane, select **Signal maps** and select the signal map you want to work with.
2. Verify that you are in the **Configure signal map** view. For more information about changing views, see: Viewing signal maps
3. In the upper-right of the signal map page, select the **Actions** dropdown menu. Select **Rediscover**.
4. You will be presented with the rediscovery screen. Select a resource that is a part of the workflow you are rediscovering. Select the **Rediscover** button.
5. The signal map will be rebuilt according to the current workflow. If you need to redeploy monitoring resources, stay on this signal map's page. Any previously attached monitoring templates will remain attached, but will need to be redeployed.

To redeploy monitoring templates after a signal map rediscovery

1. After the rediscovery, you will be directed to the updated signal map. To redeploy the monitoring templates, select **Deploy monitor** from the **Actions** dropdown menu.
2. You will be asked to confirm the deployment and presented with the number of any CloudWatch and EventBridge resources that will be created. If you would like to proceed, select **Deploy**.
3. The status of the deployment is displayed next to the name of the signal map. After a few moments of resource creation and deployment, your signal map monitoring will begin.

Deleting the signal map of your AWS media workflow

If you not longer need a signal map, it can be deleted. If you have monitoring templates deployed on the signal map, the deletion process will ask you to delete any CloudWatch and EventBridge resources that have been deployed to this signal map. Deleting the deployed resources does not affect the templates that created them. This resource deletion is to ensure that you do not have CloudWatch and EventBridge resources that are deployed but not used.

To delete a signal map

1. From the workflow monitor console's navigation pane, select **Signal maps** and select the radio button next to the signal map you want to delete.
2. Select the **Delete** button. You will be asked to confirm the deletion of the monitoring resources. Select **Delete** to begin the monitoring resource deletion process.
3. The **Monitor deployment** column will display the current status. When the status has changed to **DELETE_COMPLETE**, select the **Delete** button again.
4. You will be asked to confirm deletion of the signal map. Select **Delete** to proceed and delete the signal map.

Workflow monitor quotas

The following section contains quotas for workflow monitor resources. Each quota is on a "per account" basis. If you need to increase a quota for your account, you can use the AWS Service Quotas console to request an increase, unless otherwise noted in the following table.

Quotas

Resource type	Quota
CloudWatch alarm template groups	20
CloudWatch alarm templates	200
EventBridge rule template groups	20
EventBridge rule templates	200
Signal maps	30

Resource type	Quota
Signal maps: CloudWatch alarm template groups attached to a single signal map	5 You cannot increase this quota.
Signal maps: EventBridge rule template groups attached to a single signal map	5 You cannot increase this quota.

Using workflow monitor

Use the **overview** and **signal maps** sections of the workflow monitor console to review the current status of the workflows and any associated alarms, metrics, and logs.

Topics

- Workflow monitor overview
- Overview logs and metrics for workflow monitor
- Using workflow monitor signal maps

Workflow monitor overview

The **Overview** section of the workflow monitor console is a dashboard that provides at-a-glance information about your signal maps. In the overview section, you can see the current state of each signal map's monitoring, as well as CloudWatch metrics and any associated CloudWatch logs. You can select any signal map to be taken to that signal maps console page.

Overview filtering

Using the **Search** bar in the overview section, you can filter the list of signal maps using context sensitive constraints. After selecting the search bar, you will be presented with a list of **Properties** to filter by. Selecting a property will present **Operators** such as Equals, Contains, Does not equal, and Does not contain. Selecting an operator will create a list of resources from the selected property type. Selecting one of these resources will cause the signal map list to only display signal maps that fit the constraint you defined.

Overview logs and metrics for workflow monitor

To view CloudWatch metrics and logs for a signal map, select the radio button next to the name of the signal map. A tabbed interface for both metrics and logs will appear beneath the signal map list.

CloudWatch Metrics

CloudWatch metrics for the selected signal map will be context-sensitive and only display metrics associated with the services used in that signal maps workflow. You can use the on-screen metrics tools to customize the displayed metric periods and time ranges.

CloudWatch Logs

If you associated a CloudWatch log group with the signal map, that group will be displayed here.

Using workflow monitor signal maps

From the **overview** section of the console, you can select a specific signal map to view more information about that signal map and its attached monitoring resources.

After selecting a signal map, you will be presented with the signal map and a number of tabbed section containing more information:

- CloudWatch alarms
- EventBridge rules
- AWS Elemental alerts
- Metrics
- Logs
- Basic details

Navigating the signal map

A signal map will contain nodes for every supported AWS resource discovered by workflow monitor. Certain resources, such as MediaLive channels and MediaPackage endpoints can display thumbnail previews of the content, if thumbnail previews are available.

Selecting a resource node, and selecting **View selected resource details** from the **Actions** dropdown menu will take you to the associated service's details page. For example, selecting a

MediaLive channel and selecting **View selected resource details** will open the MediaLive console's details page for that channel.

Selecting a resource node will filter the list of active alarms to only that node. If you select the resource's **Target ARN** in the active alarm, you will be taken to the associated service's details page, with the selected resource open.

Quotas in AWS Elemental MediaTailor

MediaTailor resources and operations requests are subject to the following quotas (formerly referred to as "limits").

You can use the AWSService Quotas service to view quotas and request quota increases for MediaTailor, as well as many other AWS services. For more information, see the Service Quotas User Guide.

Quotas on ad insertion

The following table describes the quotas on AWS Elemental MediaTailor ad insertion. Unless otherwise noted, the quotas are not adjustable.

Name	Default quota value	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	5	The maximum depth of redirects that MediaTailor follows in VAST wrapper tags. MediaTailor gives up if there are additional redirects.
Ad decision server (ADS) timeout	3	The maximum number of seconds that MediaTailor waits before timing out on an open connection to an ad decision server (ADS).

Name	Default quota value	Description
		When a connection times out due to no response from the ADS, MediaTailor is unable to fill the ad avail with ads.
Ad Insertion Requests	10,000	The maximum requests per second to make for personalized manifests when performing server side ad insertion. Ad insertion handles incoming requests for manifests, session initialization, tracking data, and ad segments. This quota is adjustable.
Configurations	1,000	The maximum number of configurations that MediaTailor allows.
Content origin length	512	The maximum number of characters in a content origin specification.

Name	Default quota value	Description
Content origin server timeout	2	The maximum number of seconds that MediaTail or waits before timing out on an open connectio n to the content origin server when requesting template manifests. Timeouts generate `HTTP 504 (GatewayT imeoutExc eption)` response errors.
Manifest size	2	The maximum size, in MB, of any origin playback manifest. To ensure that you stay under the quota, use `gzip` to compress your input manifests into MediaTailor.
Package configura tions	5	The maximum number of package configurations per source (whether live or video on demand).

Name	Default quota value	Description
Prefetch schedules	25	The maximum number of active prefetch schedules per playback configuration. Expired prefetch schedules don't count towards this limit.
Server side reporting beacon request timeout	3 seconds	The maximum number of seconds that MediaTailor waits before timing out on an open connection to the server when firing a beacon for server side reporting. When a connection times out, MediaTailor is unable to fire the beacon and the service logs an `ERROR_FIRING_BEACON_FAILED` message to the *MediaTailor/AdDecisionServerInteractions* log in CloudWatch.

Name	Default quota value	Description
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.

Quotas on channel assembly

The following table describes the quotas on AWS Elemental MediaTailor channel assembly. Unless otherwise noted, the quotas are adjustable.

Name	Default quota value	Description
Channel manifest requests, per account	400	The maximum number of egress manifest requests per second for all Channel Assembly channels on an account.
Channel manifest requests, per channel	50	The maximum number of egress manifest requests per

Name	Default quota value	Description
		second for any single Channel Assembly channel.
Channel outputs	5	The maximum number of outputs per channel.
Channels per account	100	The maximum number of channels per account.
Live sources	50	The maximum number of live sources for the source location.
Programs per channel	400	The maximum number of programs per channel.
Segment delivery configurations	5	The maximum number of segment delivery configura tions per source location.
Source locations	50	The maximum number of source locations per account.
VOD Sources	1,000	The maximum number of video on demand (VOD) sources for the source location.

The following table describes the throttling limits on AWS Elemental MediaTailor channel assembly. Unless otherwise noted, the quotas are adjustable.

Name	Default transactions-per-second maximum limit	Description	
ConfigureLogsForChannel	1	Configure logs for the channel.	
CreateChannel	1	Create a channel.	
CreateLiveSource	1	Create a live source.	
CreateProgram	3	Create a program.	
CreateSourceLocation	1	Create a Source Location.	
CreateVodSource	1	Create a VOD source.	
DeleteChannel	1	Delete a channel.	
DeleteChannelPolicy	1	Delete a channel policy.	
DeleteLiveSource	1	Delete a live source.	
DeleteProgram	3	Delete a program.	
DeleteSourceLocation	1	Delete a source location.	
DeleteVodSource	1	Delete a VOD source.	
DescribeChannel	5	Describe a channel.	
DescribeLiveSoure	5	Describe a live source.	
DescribeProgram	5	Describe a program.	

Name	Default transactions-per-second maximum limit	Description	
DescribeSourceLocation	5	Describe a source location.	
DescribeVodSource	5	Describe a VOD source.	
GetChannelPolicy	5	Get a channel policy.	
GetChannelSchedule	5	Get a channel schedule.	
ListAlerts	5	List alerts.	
ListChannels	5	List channels.	
ListLiveSources	5	List live sources.	
ListPrograms	5	List programs.	
ListSourceLocations	5	List source locations.	
ListTagsForResource	5	List tags for a resource.	
ListVodSources	5	List VOD sources.	
PutChannelPolicy	3	Put a channel policy.	
StartChannel	1	Start a channel.	
StopChannel	1	Stop a channel.	
TagResource	1	Tag a resource.	
UntagResource	1	Untag a resource.	
UpdateChannel	1	Update a channel.	

Name	Default transactions-per-second maximum limit	Description
UpdateLiveSource	1	Update a live source.
UpdateProgram	1	Update a program.
UpdateSourceLocation	1	Update a source location.
UpdateVodSource	1	Update a VOD source.

AWS Elemental MediaTailor resources

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

Resource	Description
SCTE standard: SCTE 35	The SCTE standard document for SCTE35.
Classes and workshops	Links to role-based and specialty courses as well as self-paced labs to help sharpen your AWS skills and gain practical experience.
AWS developer tools	Links to developer tools, SDKs, IDE tool kits, and command line tools for developing and managing AWS applications.
AWS whitepapers	Links to a comprehensive list of technical AWS whitepapers, covering topics such as architect ure, 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 concernin g AWS billing, account, events, abuse, and other issues.

Resource	Description
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.

Change	Description	Date
Added vended logs information	Added new section about using vended logs for publishing logs that MediaTailor emits..	February 4, 2025
Added preconditioned ads overview	Added new section that describes how MediaTailor manages ad insertion when you're using preconditioned ads.	January 30, 2025
Added ad conditioning	Added the **Streaming media file conditioning** setting.	January 30, 2025
Added log types	Added new section that lists the log types that MediaTailor emits.	January 15, 2025
Added log filtering information	Added new section about filtering logs that MediaTailor emits.	January 15, 2025
Added supported query parameter formatting	Added sections for manifest query parameter and ADS query parameter formatting.	January 2, 2025
Integration information for Google Ad Manager	Added a section about integrating MediaTailor with Google Ad Manager from server side and client side.	November 25, 2024

Updated variable	Changed `breakabilityStartTime` to `availabilityStartTime`.	May 6, 2024
Added personalization detail	Added the `Insertion Mode` personalization detail.	May 6, 2024
Program Rules	Added new content on program rules.	April 25, 2024
Updated content on creating channels	Added information about program rules when creating channels.	April 20, 2024
Updated content on adding programs	Added information about program rules when adding programs.	April 20, 2024
Workflow monitor	Analyze AWS media services and create signal maps, visualizations of the media workflow, between those services. Use the signal maps to generate monitoring alarms and notifications using CloudWatch, EventBridge, and AWS CloudFormation.	April 11, 2024
AlternateMedia and the As Run Log	Added a note on how AlternateMedia affects the As Run Log.	February 28, 2024
Time-shifted viewing	MediaTailor channels support time-shifted viewing for content that's up to 6 hours old.	December 27, 2023

Updated manifest settings	Added information about passthrough tags, depending on the chosen Ad markup type.	November 28, 2023
SCTE-35 messages for ad breaks	Added information about inserted SCTE-35 tags for `Daterange` versus `Scte35 Enhanced` Ad markup types.	November 28, 2023
Key-value pairs for the `Enhanced Scte35` Ad markup type	Added information about how MediaTailor handles submitted key-value pairs for the `Enhanced Scte35` Ad markup type.	November 28, 2023
VOD source ad opportunities	MediaTailor can now automatically detect ad opportunities on VOD sources.	October 6, 2023
New Autodetect SigV4 authentication type	MediaTailor now supports the `AUTODETECT_SIGV4` access type.	August 18, 2023
Updated client-side tracking content	Updated the client-side tracking content to include additional information.	August 12, 2023
Setting up MediaTailor and MediaPackage workflows to use live sources	Added information on setup, general requirements, and behavior when MediaTail or and AWS Elemental MediaPackage workflows use live sources.	May 24, 2023

Manifest query parameters documentation	Added a section that describes manifest query parameters.	April 26, 2023
Overlay ads documentation	Added a section that describes overlay ads.	April 24, 2023
Ad ID decoration in manifests documentation	Added a section that describes ad ID decoration in manifests.	April 24, 2023
Added AFTER_LIVE_EDGE suppression mode	`AFTER_LIVE_EDGE` ad suppression mode is now available in addition to `BEFORE_LIVE_EDGE` mode.	February 21, 2023
New As Run log	New topic on the As Run log.	January 19, 2023
IAM best-practices updates	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM.	December 27, 2022
IAM best-practices updates	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM.	December 27, 2022
Updated quotas content	Updated and reorganized quotas information.	September 13, 2022
Added and corrected channel assembly quotas	Added quotas for live sources, segment delivery configura tions, manifest requests, and channel transactions in the MediaTailor channel assembly service.	September 13, 2022

Added and corrected channel assembly quotas	Added quotas for logging, channel, live sources, programs, source locations , and channel policies in the MediaTailor channel assembly service.	September 11, 2022
New channel assembly alerts table	You can now see a table that explains channel assembly alerts.	September 1, 2022
New Amazon CloudWatch metrics	Added new CloudWatch metrics.	June 26, 2022
Ad calls topic	Added link to prefetch support for VAST responses.	May 25, 2022
New ADS request variables	MediaTailor now supports these additional SCTE-35 variables in ADS requests: `scte.segmentation_type_id` , `scte.avails_expected` , `scte.delivery_not_restricted_flag` , `scte.segment_num` , `scte.sub_segment_num` , `scte.segments_expected` , `scte.sub_segments_expected` , `scte.device_restrictions` , `scte.no_regional_blackout_flag` , `scte.archive_allowed_flag` and `scte.segmentation_event_id` .	April 27, 2022

New IAM managed policy policy topic	Added two new managed policies for MediaTailor.	November 24, 2021
New `AWSElementalMediaTailorReadOnly` managed policy	Added a new AWS managed policy that grants permissions that allows read-only access to MediaTailor resources.	November 10, 2021
New `AWSElementalMediaTailorFullAccess` managed policy	Added a new AWS managed policy that allows full access to MediaTailor resources.	November 10, 2021
New confused deputy topic	Added a topic that explains how to prevent the confused deputy problem.	November 4, 2021
Prefetching ads topic	MediaTailor can now prefetch ads for ad breaks before they occur.	October 12, 2021
Added logging configuration settings for playback configurations	Use logging configuration settings to control settings related to playback configuration logs.	September 28, 2021
SCTE-35 messages for ad breaks	Added information about using `segmentation_descriptor` messages together with `time_signal` messages.	September 1, 2021
New linear playback mode	Added a new linear playback mode.	September 1, 2021

New absolute transition type	Added support for absolute transition types, which you can use to set a wall clock start time for your program on linear channels.	September 1, 2021
New channel assembly alerts topic	You can now monitor your channel assembly resources using MediaTailor alerts. When an issue or a potential issue occurs with your channel assembly resources, MediaTail or generates alerts.	July 14, 2021
Corrected channel assembly quotas for channels egress requests	Corrected quotas for channels egress requests in the MediaTailor channel assembly service.	June 29, 2021
New source location authentication type	MediaTailor now supports Secrets Manager access token authentication.	June 16, 2021
New Tier information	Added information about the modes and source types that each Tier supports.	June 13, 2021
New Source type information	For Standard channels, added information about the type of source that the program plays.	June 13, 2021

New MediaTailor live sources documentation	A live source represents a single live stream that you add to your source location. After you create your channel, you can add live sources to your source location and associate each live source with a program.	June 13, 2021
Support for additional UPID types	MediaTailor now supports ADS Information (0xE) and User Defined (0x1) segmentation UPID types.	April 15, 2021
New segmentation UPID dynamic variables	There are three new dynamic variables: `scte.segmentation_upid.asseId` , `scte.segmentation_upid.cueData.key` , and `scte.segmentation_upid.cueData.value` . These variables are used in conjunction with the MPU segmentation UPID type (0xC) for podbuster workflows.	April 15, 2021
New channel assembly service description	Added information about the new channel assembly service.	March 11, 2021
New MediaTailor channel assembly service documentation	Channel assembly is a new manifest-only service that allows you to create linear streaming channels using your existing video on demand (VOD) content.	March 11, 2021

Added channel assembly quotas	Added quotas for the new MediaTailor channel assembly service.	March 11, 2021
New channel assembly terms	Added terms that correspond to the new channel assembly service.	March 10, 2021
Channel assembly tagging support	Added support for tagging of channel assembly resources in AWS Elemental MediaTail or. Channels, SourceLocations, and VodSources support tagging.	March 9, 2021
New dynamic variables topic	MediaTailor now supports dynamic domain variables.	February 25, 2021
Added optional configuration alias settings	Use configuration aliases alongside domain variables to dynamically configure domains during session initialization.	February 25, 2021
New `scte.segmentation_upid` dynamic ad variable	Added support for the `scte.segmentation_upid` session data dynamic ad variable.	December 5, 2020
New ad marker passthrough topic	Ad marker passthrough is now available for HLS manifests.	October 29, 2020
Updated configuration advanced settings	Ad marker passthrough is a new playback configuration advanced setting.	October 14, 2020
New debug log mode	New topic on the DEBUG log mode.	August 14, 2020

Clarification around EXT-X-CUE-OUT duration attribute for bumpers	Updated the bumpers requirements so that for HLS the `duration` attribute is required for each `EXT-X-CUE-OUT` tag.	August 5, 2020
New bumpers topic	Added a new bumpers topic	July 27, 2020
Ad Suppression is available for DASH	Ad suppression is now available for DASH. Removed the "HLS-only" restriction from the ad suppression topic.	June 3, 2020
Update console-specific names	Updated console-specific names to reflect a newer version of the console UI.	May 1, 2020
New `avail.index` dynamic ad variable	Added support for the new `avail.index` session data dynamic ad variable.	March 13, 2020
New `AdVerifications` and `Extensions` elements	For client-side reporting, the `AdVerifications` and `Extensions` elements are supported.	March 10, 2020
Personalization threshold configuration	Added support for the personalization threshold optional configuration.	February 14, 2020
DASH VOD manifests	Added support for video on demand (VOD) DASH manifests from the origin server, with multi-period manifest output.	December 23, 2019

Console support for transcode profile name	Added description for transcode profile name in the configuration.	December 23, 2019
Updated limits tables	Updated limits for ADS redirects and ADS timeouts.	December 18, 2019
CDN best practices	Added a section about content distribution network (CDN) best practices for personalized manifests.	December 13, 2019
Document live pre-roll behaviors	Added *Pre-Roll Ad Insertion* section to describe how live pre-roll ads work with AWS Elemental MediaTailor.	November 26, 2019
Support for live pre-roll ads	Added support for inserting pre-roll ads at the beginning of a live stream.	September 11, 2019
Analyzing ADS logs in Amazon CloudWatch Logs insights	Added information for using the AWS Elemental MediaTail or ADS logs and CloudWatch Logs Insights to analyze your MediaTailor sessions.	August 13, 2019
New security chapter	Added a security chapter to enhance and standardize coverage.	May 23, 2019
DASH single-period manifests	Added support for single-pe riod DASH manifests from the origin server, with multi-per iod manifest output.	April 4, 2019

Support for SCTE-35 UPIDs in the ADS URL	Added support for including a unique program ID (UPID) in the ad decision server (ADS) URL. This allows the ADS to provide program-level ad targeting within a live linear stream.	March 28, 2019
Client-side reporting supports companion ads	For client-side reporting, the AWS Elemental MediaTail or tracking URL response now includes companion ad metadata.	March 28, 2019
HLS ad marker documentation	Added a section that describes supported HLS ad markers.	March 1, 2019
Tagging support	Added support for tagging of configuration resources in AWS Elemental MediaTail or. Tagging allows you to identify and organize your AWS resources and control access to them and to track your AWS costs.	February 14, 2019
Added AWS CloudTrail logging information	Added topic about using CloudTrail to log actions in the AWS Elemental MediaTail or API.	February 11, 2019

Added section on playback errors	Added information about the errors that might be returned by MediaTailor during playback, in response to requests from a player or a content delivery network (CDN).	February 4, 2019
DASH base64-encoded binary	Added support for providing splicing information in manifests in base64-encoded binary, inside `<scte35:Signal>` `<scte35:Binary>` markers.	January 4, 2019
DASH time signal	Added support for providing splicing information in manifests inside `<scte35:TimeSignal>` markers.	December 5, 2018
DASH location support	Added support for the MPEG-DASH `<Location>` tag.	December 4, 2018
DASH support	Added support for MPEG-DASH manifests.	November 14, 2018
Updated limits tables	Updated limits for configurations and manifest size.	October 13, 2018
New and updated metrics	Added metrics for ad decision server (ADS) and origin timeouts, and updated the ADS and origin error definitions to include timed-out responses.	October 13, 2018

Better documentation coverage for server-side and client-side ad insertion use cases	Expanded description and examples to cover the use of dynamic ad variables for server-side ad insertion and for client-side ad insertion.	October 1, 2018
New regions	Added support for the PDX and FRA Regions.	July 18, 2018
VAST/VPAID	Added information about VAST and VPAID.	March 16, 2018
CloudWatch	Added information about available CloudWatch metrics, namespaces, and dimensions.	March 16, 2018
New regions	Added support for the Asia Pacific (Singapore), Asia Pacific (Sydney), and Asia Pacific (Tokyo) Regions.	February 8, 2018
Default Amazon CloudFront distribution paths	Added the list of paths for the Amazon CloudFront distribution where AWS Elemental MediaTailor stores ads.	February 6, 2018
IAM policy information	Added IAM policy information specific to AWS Elemental MediaTailor. Added instructions for creating non-admin roles with limited permissions.	January 3, 2018
First release	First release of this documentation.	November 27, 2017

Note

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