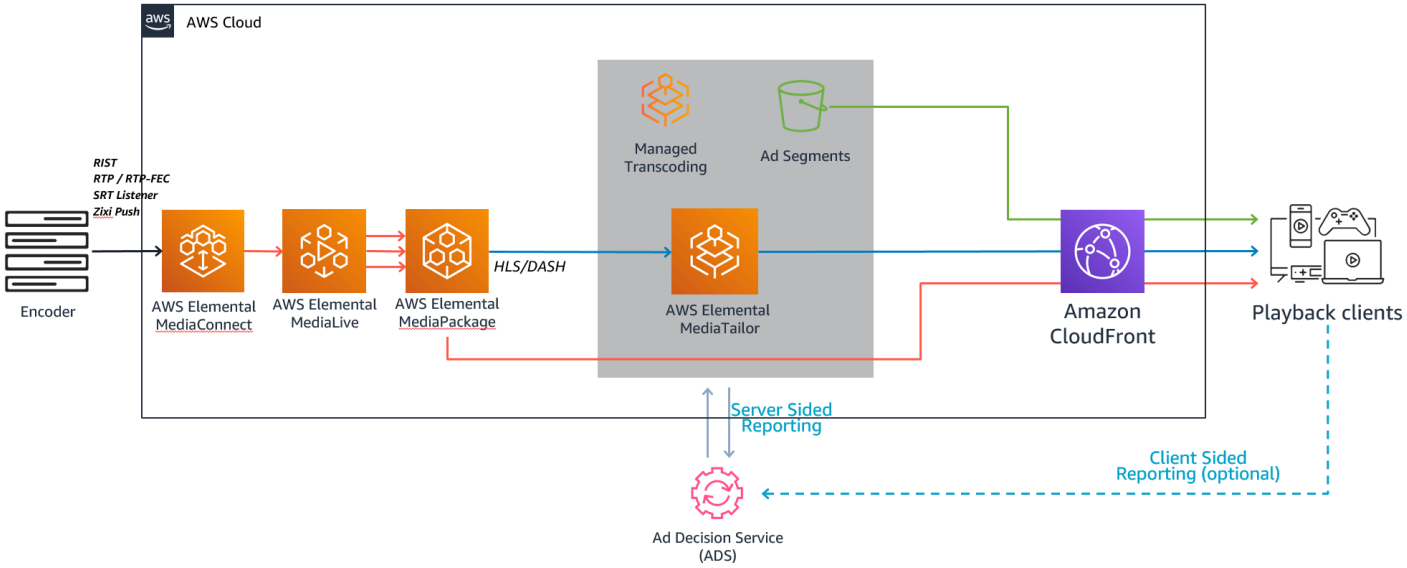
How to implement a reliable DAI solution using AWS Media Services

Content

Workflow overview

AWS Elemental MediaTailor allows you to serve targeted ads to viewers while maintaining broadcast quality in over-the-top (OTT) video applications. This step-by-step article explains how to build a reliable Live DAI on AWS MediaServices. Here is the architecture that is discussed in this guide:



- **Ingestion:** The source can be any live feed using SCTE markers. In this blog article, we will be using a live source from AWS Elemental MediaConnect—a high-quality transport service for live video. The source is an MPEG-2 transport stream that has SCTE-35 markers and includes multiple components: audio, video, and subtitles.
- **Live transcoding:** We will use MediaLive, which works as a live transcoder to create multiple video renditions. The SCTE markers can be pass-throughs from the source or can be directly sent to MediaLive through an API call (using the schedule feature).
- **Packaging:** We will use AWS Elemental MediaPackage—which reliably prepares and protects video for delivery over the internet—as a packager and to convert the stream on the fly from MediaLive into various formats: HTTP Live Streaming (HLS) and MPEG Dynamic Adaptive Streaming over HTTP (DASH).
- **SSAI:** We will use MediaTailor as a manifest manipulator. It will stitch the ads received from the ad decision service (ADS) to the live stream. The ADS can be any ad server for direct sold advertising and includes the supply-side solutions for programmatic sold advertising that are downstream from the ad server.
- **CDN:** We will use Amazon CloudFront as the CDN to distribute the live content. Note that you can also use any other third-party CDNs.
- **Player:** You will need to use a video player supporting HLS with the “EXT-X-DISCONTINUITY” tag or a video player supporting MPEG-DASH with multiple periods. MPEG-DASH streams with single periods are also supported by MediaTailor as origins (more info here).

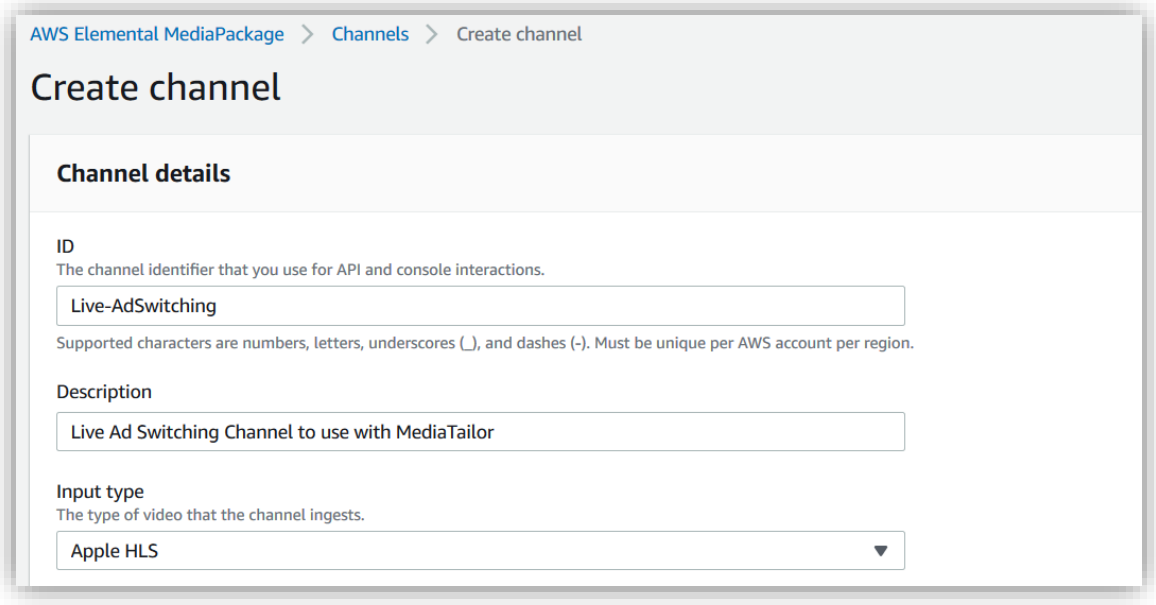
It is worth noting that MediaTailor will also support other non-AWS-origin and CDN solutions, but we are presenting a complete AWS configuration here.

Step 1: Configure AWS Elemental MediaPackage

MediaPackage is a just-in-time video packaging and origination service that runs on AWS. With MediaPackage, you can deliver highly secure, scalable, and reliable video streams to a wide variety of playback devices and CDNs.
MediaPackage can create HLS, MPEG-DASH, Common Media Application Format (CMAF), and even Microsoft Smooth Streaming outputs using just-in-time packaging. MediaPackage supports SCTE-35 messaging (more info here).
Let's first create a MediaPackage channel because you will need the MediaPackage channel identifier (ID) later as the output of the MediaLive channel.

Creating an AWS Elemental MediaPackage channel

- Go to the MediaPackage landing page.
- To create a channel, please follow the documentation here.

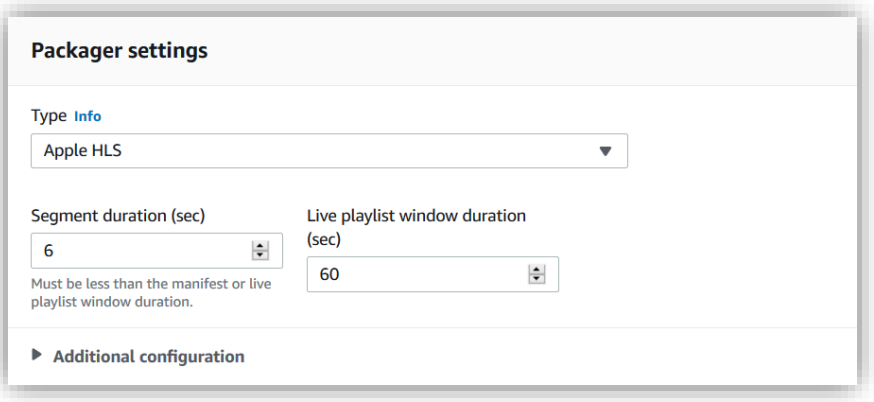


- You can activate **access logging** to publish all incoming or outcoming requests on MediaPackage (more information here).
- Select the option **Create a CloudFront distribution for this channel**.

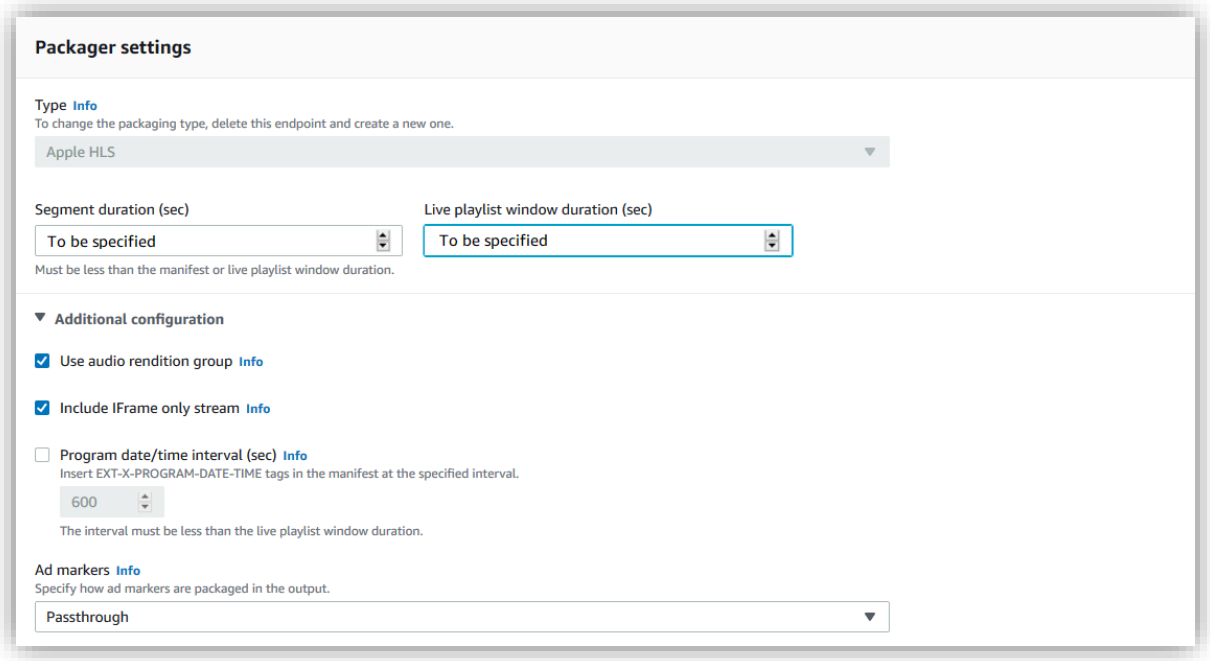
Once the MediaPackage channel is created, edit this channel and create endpoints for HLS and MPEG-DASH using the following directions.

Creating an HLS endpoint

- In the **Endpoints** menu, click **Add endpoints**.
- When creating the endpoint, in the **Packager settings**, select the type **Apple HLS** and your desired segment duration.



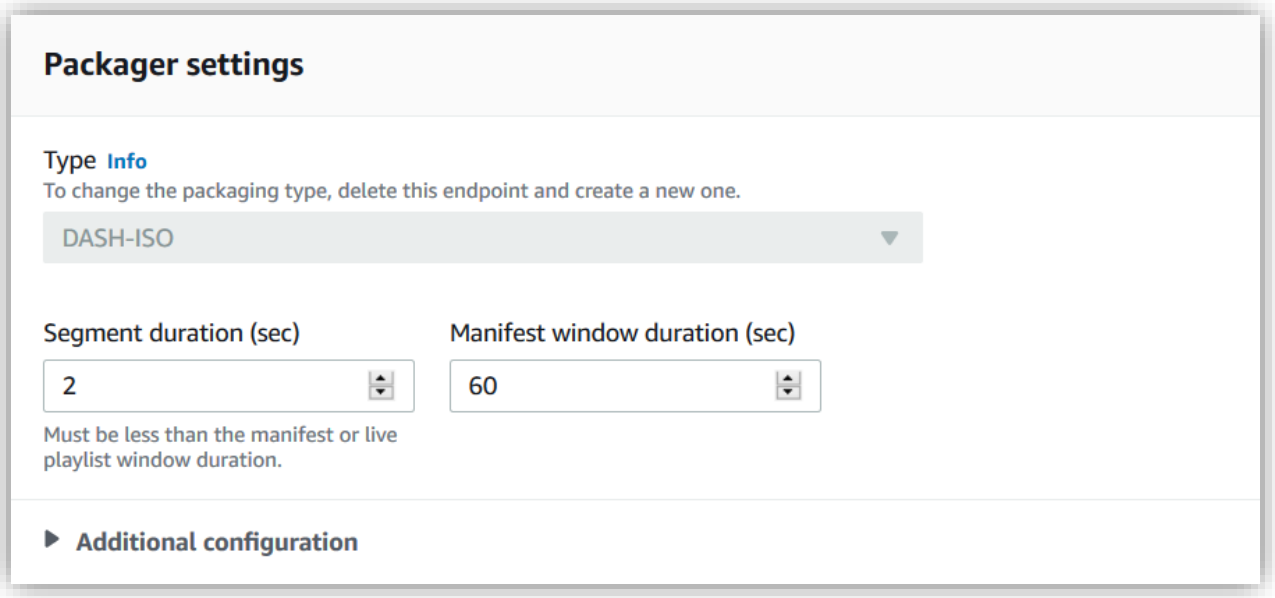
- In **Additional configuration**, make sure to configure the ad markers in the output of MediaPackage. There are various modes available on MediaPackage, but we will use only the **Passthrough** option. Further documentation on SCTE-35 markers with MediaPackage can be found here.
- The option **Use audio rendition group** will create an HLS stream with separate audio and video chunks (known as demultiplexed mode).



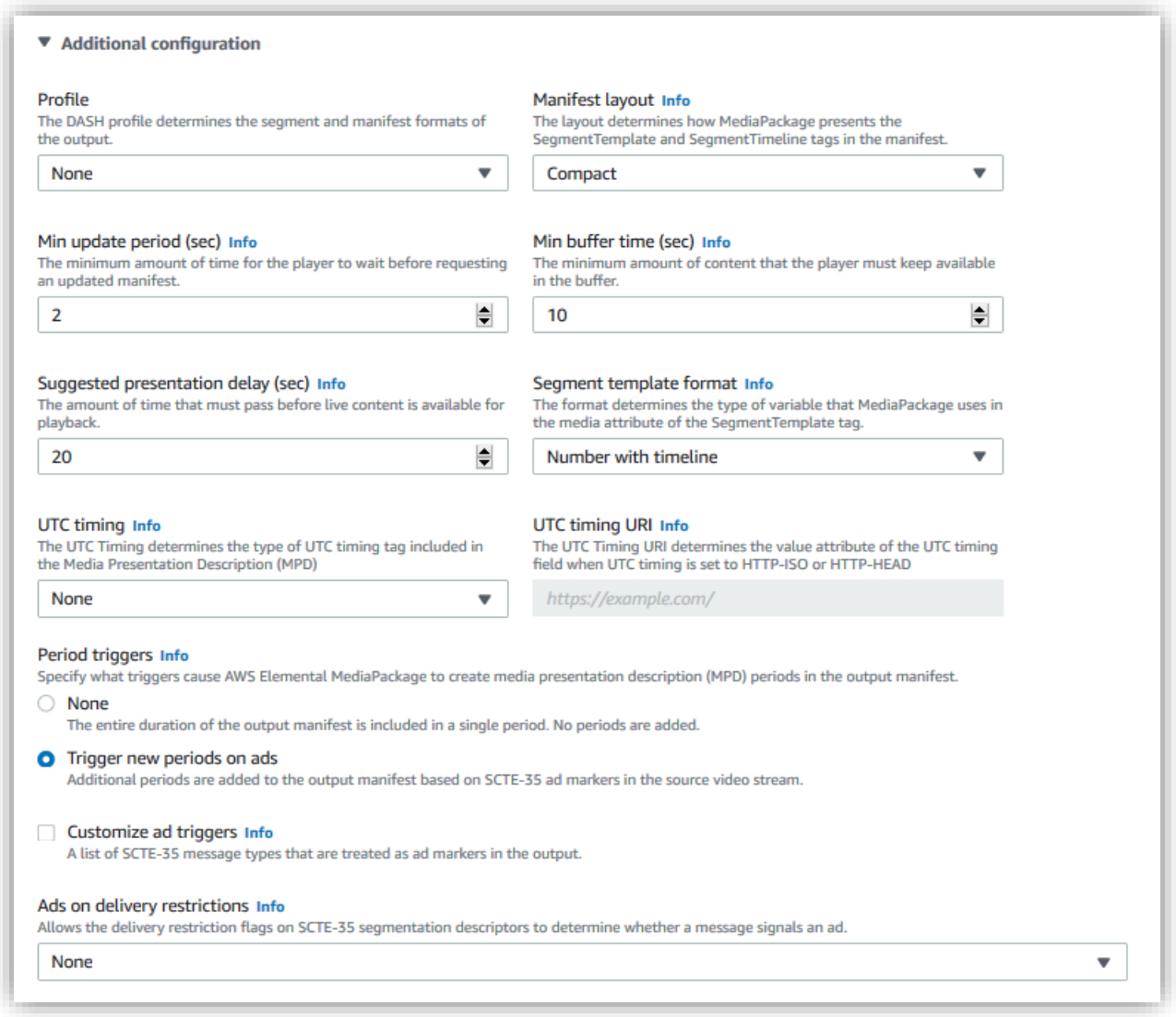
- You can secure your HLS MediaPackage endpoint using the **Access control settings** to limit who can access your HLS endpoint (more documentation here).

Creating an MPEG-DASH endpoint

- Now, let's add a new endpoint in MPEG-DASH. Select the type **DASH-ISO** with your desired segment duration in the **Packager settings**.



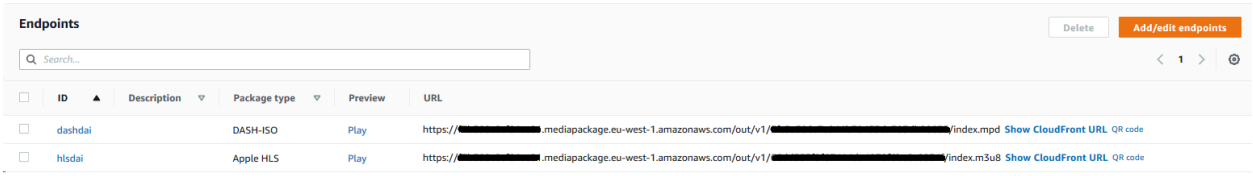
- In **Additional configuration**, specify the DASH profile, template to be used, and more. Further documentation on creating a DASH packaging configuration can be found here.
- Make sure to configure the ad markers in the output of MediaPackage. Choose **Trigger new periods on ads** to get a multiperiod DASH manifest during ad breaks.



- You can secure your DASH MediaPackage endpoint using the **Access control settings** to limit who can access your HLS endpoint (more documentation here).

Saving URLs for later use

- Go back to your channel details, and you'll be able to get the URLs for your HLS and DASH endpoints.



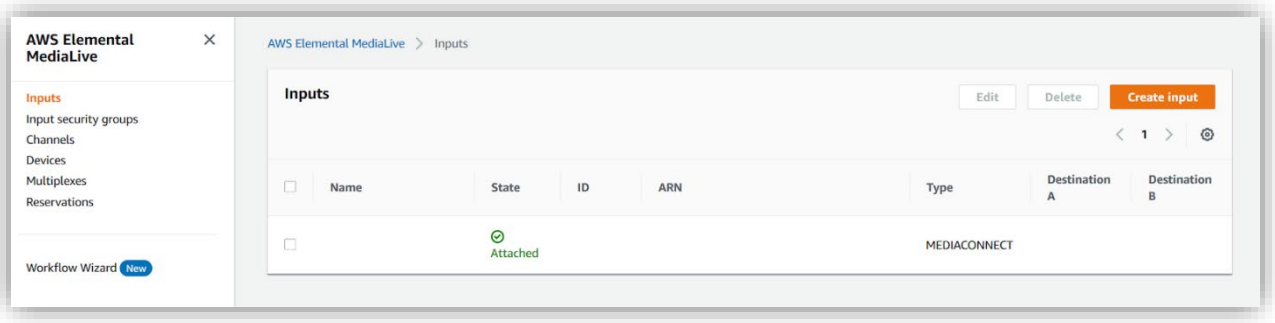
- The format is the following:
  - HLS: https://playback-endpoint-EMP/out/v1/hashed-id-hls/index.m3u8
  - MPEG-DASH: https://playback-endpoint-EMP/out/v1/hashed-id-dash/index.mpd
- You will need these URLs later to configure MediaTailor and Amazon CloudFront.

Step 2: Configure AWS Elemental MediaLive

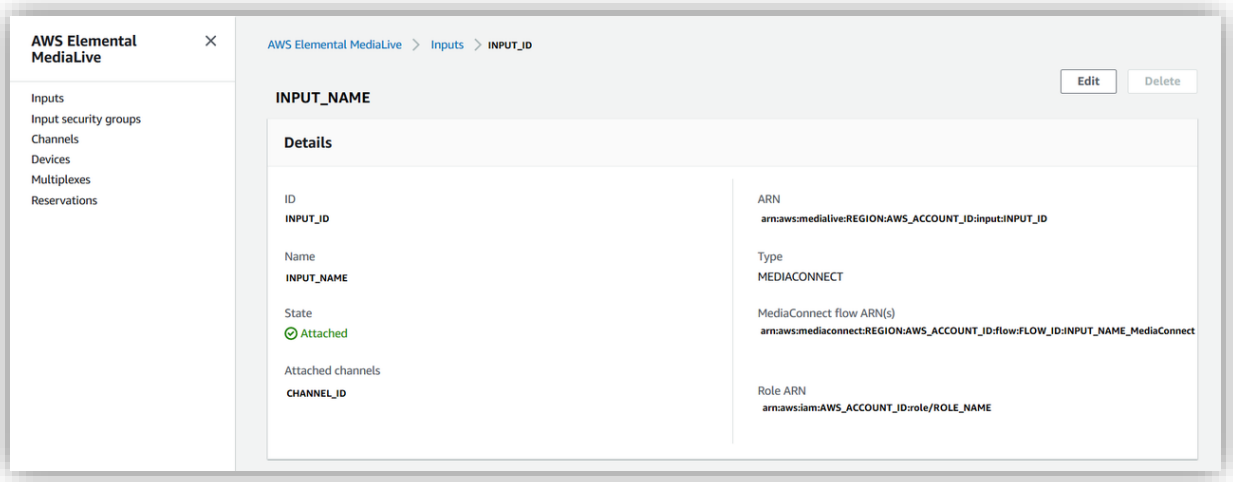
MediaLive is a near-real-time video service that lets you create live outputs for broadcast and streaming delivery. MediaLive will generate the different renditions for the adaptive bitrate (ABR) stack and push it to MediaPackage.
In this example, we will be using an MPEG-2 transport stream source—including multiple tracks (video, audio, and subtitle tracks) and SCTE-35 markers—by using a MediaConnect input.

Creating an AWS Elemental MediaLive input

- Go to the MediaLive landing page, and choose the same AWS Region as your MediaPackage channel's Region.
- Select the **Inputs** menu on the left pane, and click **Create input**.



- We'll be using a MediaConnect input with SCTE-35 markers as the source for MediaLive. For further information about MediaLive, please read the documentation. This is an example of what a MediaConnect input would look like on MediaLive inputs:

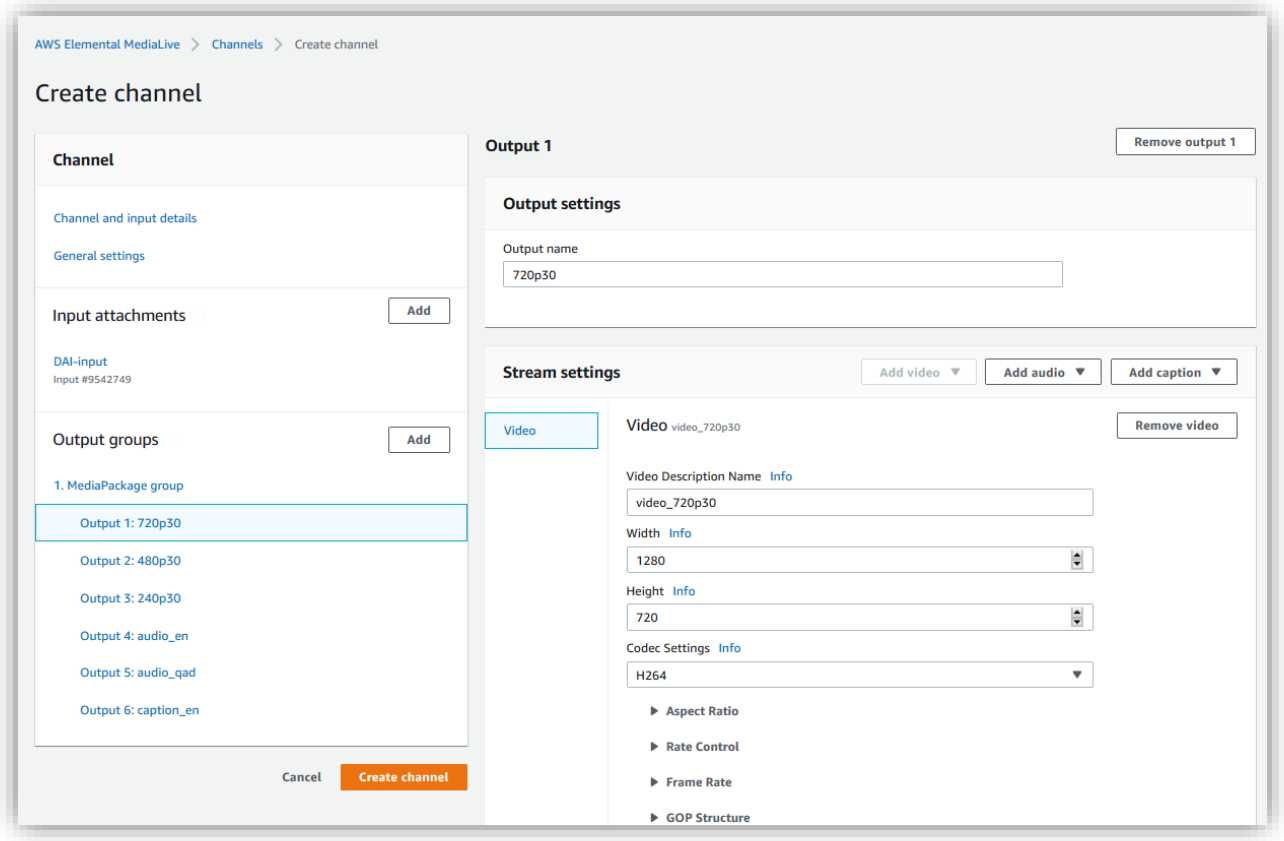


NOTE: You can use any source from the supported input list (more documentation on creating an input here). You can initiate an SCTE-35 marker directly on MediaLive using the schedule feature. You can easily send a marker to MediaLive using the schedule feature on the console user interface (more info here). You can also programmatically send the marker through the API (more information here).

Creating an AWS Elemental MediaLive channel

- Now, go back to the **Channels** menu on the left pane, and click **Create channel**.
- In the channel settings page, there are various parameters to configure your MediaLive channel (for example, channel details, inputs, codecs, and outputs). Here are some important settings to configure in **Channel and input details**:
    - Select or create the correct AWS Identity and Access Management (AWS IAM) role (more info). (AWS IAM provides fine-grained access control across all of AWS.)
    - You can use a template—for example, the **Live action (MediaPackage)** template.
    - Select the channel class you want to use—either a **Standard** or **Single pipeline** channel (more info).
    - Select the right input specifications depending on your live source (for example, SD, HD, or UHD).
- Go to the **General settings** menu:
    - You can work with SCTE-35 signaling (mainly used for timed signal messaging) and activate the **Avail configuration** setting if needed (more info). (For further information about the SCTE-35 message processing with MediaLive, please read the documentation here.)
    - In the **Timecode configuration** settings, change the value for **SYSTEMCLOCK** (more info).
    - In the **Channel logging** settings, select the **DEBUG** mode to make sure you can analyze your SCTE-35 markers using Amazon CloudWatch (a monitoring and observability service) for troubleshooting. (Find out more about monitoring a channel using Amazon CloudWatch logs here.)
- Now, attach the MediaConnect input that you just created (more information about the input attachments settings here). Note that you can map the source components by specifying the right audio selectors or caption selectors. You can also provide automatic input failover (more documentation here).
- In **Output groups**, assign the MediaPackage channel that you just created in the previous section (more documentation here).
- Finally, modify the different settings for your channels depending on your requirements for each output (for example, codecs, resolution, and groups of pictures [GOPs]).

In our example, we used a demultiplexed HLS stream on the MediaPackage output to get three video renditions, two audio tracks, and one caption track. Each output will contain only one single component (more documentation about the stream sections here). Make sure you provide a **Stream name** for your different audio and caption tracks.
This is an example of a MediaLive output using demultiplexed streams:

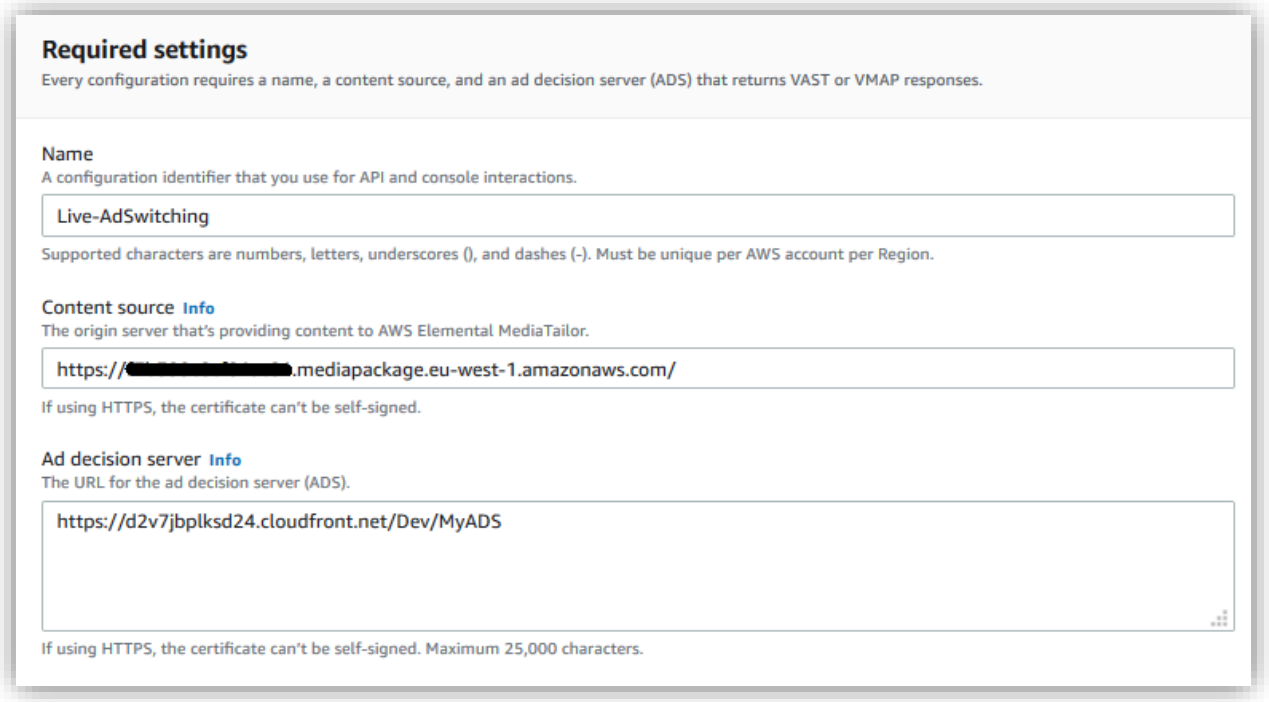


Step 3: Configure AWS Elemental MediaTailor configuration

MediaTailor is a scalable ad insertion and channel assembly service that runs on AWS. Using MediaTailor, you can serve targeted ad content to viewers and create linear streams while maintaining broadcast quality in OTT video applications. MediaTailor ad insertion supports HLS and MPEG-DASH.

Creating an AWS Elemental MediaTailor configuration

- Go to the MediaTailor landing page, and choose the closest Region to your MediaPackage Region.
- For **Content source**, enter the hostname from the MediaPackage endpoints' URLs.
- For **Ad decision server**, enter the URL of your ADS. To better personalize the answer from your ADS, you can use specific session variables as query-string parameters in your ADS URL (more info here).
- Fill out the required fields, and click **Create configuration**.

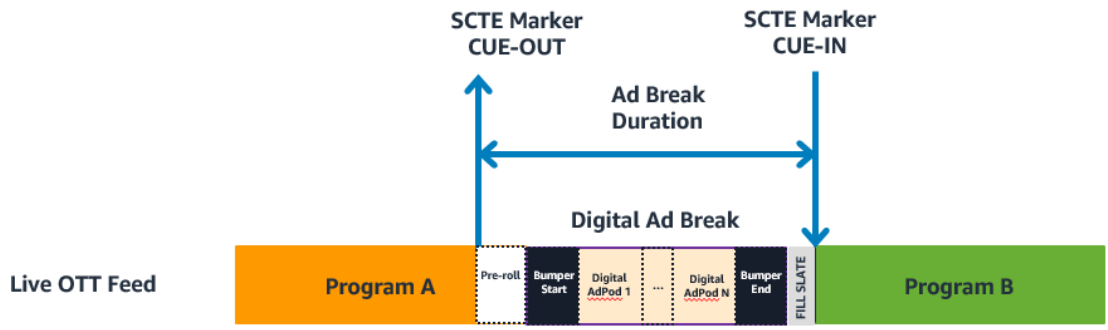


Creating a sample ad server answer

- If you don't have a third-party ad server to provide an ADS URL, you can create your own ADS. It will be an XML file stored on Amazon Simple Storage Service (Amazon S3), an object storage service.
- For testing purposes, please either go to the following blog post for further information on how to build your own Video Ad Serving Template (VAST) 3.0 response or use these VAST samples:
    - With no tracking: https://d2v7jbplksd24.cloudfront.net/Dev/MyADS
    - With tracking events: https://d11egurzincj00.cloudfront.net/TestVAST.xml

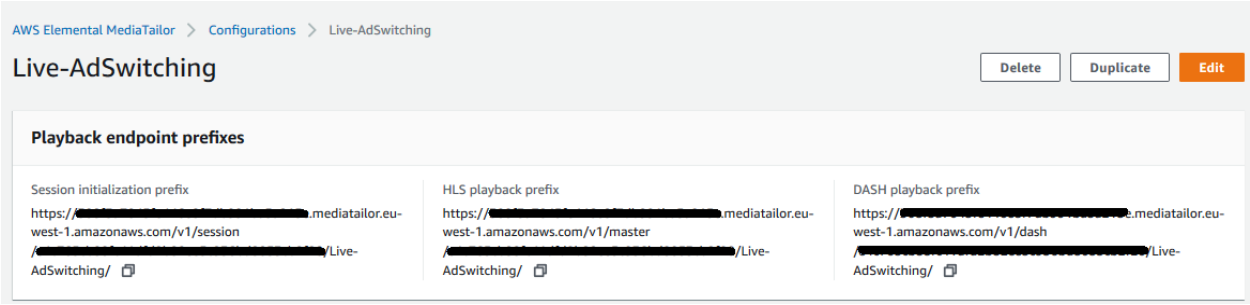
Customizing the AWS Elemental MediaTailor configuration

- MediaTailor offers multiple options to better optimize the ad replacement experience for your end user. Here are all the different personalization details you can implement:
    - Inserting a slate ad at the end of the ad pods (more info here)
    - Inserting preroll ads (more info here)
    - Removing ad insertions for midbreak joiners or long digital video recorder (DVR) window (more info here)
    - Inserting bumpers that play at the start of or before the end of an ad break (more info here)
    - Creating a custom transcoding profile with the help of AWS Support
- You can also change the hostnames used to distribute the ad segments or the manifest files using the *CDN content / ad segment prefix* in **Advanced settings**. These settings will masquerade every asset (live content and/or ad-served content) under one single CDN configuration to provide relative URLs. MediaTailor is compatible with any third-party CDN (more info here).
- Here is an illustration of what an ad break using MediaTailor would look like using the personalization features:



Saving URLs for later use

- Now that you have created the MediaTailor configuration, you can save the *playback URLs* in HLS/MPEG-DASH and the *session URL*, to initiate the session and the client-side tracking.



- To validate the output of MediaTailor, you can construct your playback URL from the MediaPackage endpoints using the following format:
    - HLS playback URL: https://playback-endpoint-EMT/v1/master/hashed-account-id/origin-id/out/v1/hashed-id-hls/master.m3u8
    - MPEG-DASH playback URL: https://playback-endpoint-EMT/v1/master/hashed-account-id/origin-id/out/v1/hashed-id-dash/manifest.mpd
- The session URL is used to initialize a session with MediaTailor and also to get the tracking URL for the client-side reporting (more info here):
    - Session initialization prefix: **https://playback-endpoint-EMT/v1/session/hashed-account-id/origin-id/**

Ad reporting

When the player first requests the DASH manifest or the HLS master URLs to MediaTailor, it initializes a unique playback session on MediaTailor. By default, MediaTailor will perform the server-side reporting by sending all the appropriate beacons to the ad server through the ad tracking URLs (more info about server-side reporting here). But it is also possible to implement a client-side reporting with MediaTailor for further functionalities, including the following:

- Supporting Video Player-Ad Interface Definition (VPAID), ClickCompanion, Open Measurement, and more
- Scrubbing behavior, where a viewer can scrub through the video by moving the cursor on the playback bar
- Advanced playback behaviors that require player development, like no-skip-forward and countdown timers on ad avails

When using client-side reporting, you will need to send a POST request to the session *initialization prefix* listed above in your MediaTailor configuration. The answer from that POST request will provide the unique playback session and the tracking URL to use for your client-side reporting. For example:
`POST https://playback-endpoint-EMT/v1/session/hashed-account-id/origin-id/out/v1/hashed-id-hls/master.m3u8`

- ⇨ HTTP answer would be:

```
{
  “manifestUrl”:”/v1/master/hashed-account-id/origin-id/out/v1/hashed-id-hls/master.m3u8?aws.sessionId=<session.id>”,
  “trackingUrl”:”/v1/tracking/hashed-account-id/origin-id/out/v1/hashed-id-hls/<session.id>”
}
```

You will need to poll the tracking URL frequently to get the JSON answer with time offsets for the ad avails. More information about the client-side reporting can be found here.

Observability and monitoring

Another great benefit of using MediaTailor is that you can easily troubleshoot what is happening with your origin and your ADS. Once you provide MediaTailor the access to log in to Amazon CloudWatch (more info here), you get a complete set of monitoring tools.

- You can create Amazon CloudWatch dashboards—customizable home pages in the Amazon CloudWatch console—in near real time using the different MediaTailor metrics (list of all the metrics here).
- You can dive deep into information about individual sessions, viewers, issues with ad servers / beacons / client devices, or ad opportunities to understand why spots were or weren’t filled (more information in this detailed blog post).
- You can also activate debug logs to trace the origin interactions, generated manifest, or initialized session (more info here).

Step 4: Configure Amazon CloudFront

Amazon CloudFront is a content delivery network, and it delivers your content through a worldwide network of data centers called “edge locations.” When a user requests content that you’re serving with Amazon CloudFront, the request is routed to the edge location that provides the lowest latency (time delay) so that the content is delivered with optimal performance to your end users.
When configuring Amazon CloudFront or any other CDNs for a DAI workflow, remember the following:

- Personalized manifests generated by MediaTailor are **NOT** cached by CDNs.
- Live content video can be cached by CDNs.
- Ad video can be cached by CDNs.

Retrieving the distribution created by AWS Elemental MediaPackage

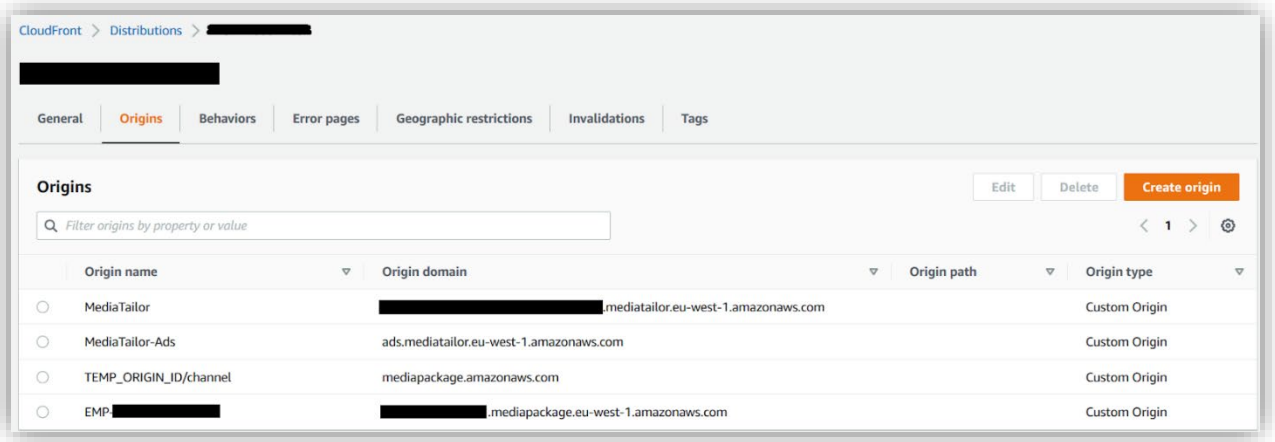
- Go to the Amazon CloudFront landing page.
- Select the distribution created by MediaPackage during step 1. You can easily find it using the **Last modified** filter with the latest distribution. (The distribution will contain the comment “Managed by MediaPackage. . . .”)

Configuring AWS Elemental MediaTailor and its ad origins

- Click on the **Origins** tab; then click **Create origin**.

- Enter the MediaTailor hostname for the **Origin domain**. (Use the playback-endpoint-EMT "*.mediatailor.eu-west-1.amazonaws.com".)
- Update the **Origin protocol policy** to **HTTPS**.
- Leave the **Origin path** blank, and update the **Origin ID** to "MediaTailor".
- Click **Create**.
- Reiterate the process to create the origin for the ad-served content.
- Enter the MediaTailor ad server hostname for the **Origin domain name**. The hostname depends on the Region: "ads.mediatailor.**REGION**.amazonaws.com" (for example, "ads.mediatailor.eu-west-1.amazonaws.com" for the AWS Europe [Ireland] Region)
- Leave the **Origin path** blank. Select **HTTPS**, and update the name to "MediaTailor-Ads".
- Click **Create**.

NOTE: If you are using a CDN authorization for your MediaPackage endpoints, you will need to add the header `X-MediaPackage-CDNIdentifier` with the secret value to make sure that MediaTailor can access your MediaPackage origin.

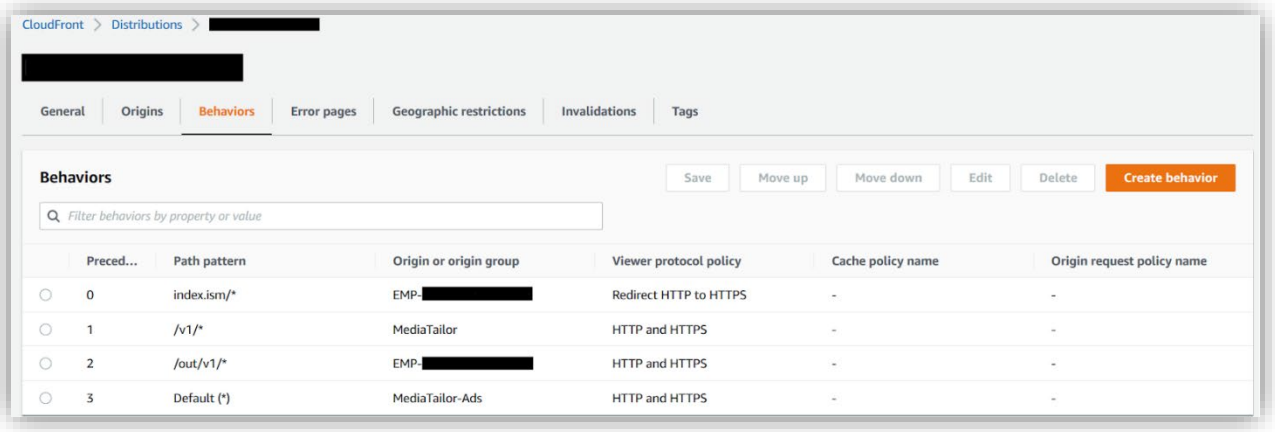


Configuring the cache behaviors to your distribution

- Now, click on the **Behaviors** tab, and click **Create behavior**.
- Select the **Default behavior**, and click **Edit**.
- Update the **Origin,** and select the **MediaTailor-Ads** origin.
- Set **Cache key and origin requests** to **Legacy cache settings** with **Headers**/**Query strings**, and **Cookies** set to **None**.
- Click **Save changes**.
- Click **Create** to add another cache behavior for the MediaTailor origin.
- Enter "/v1/*" for the **Path pattern**.
- Under **Origin**, select the **MediaTailor** origin.
- Set **Cache key and origin requests** to **Legacy cache settings** with **Headers**/**Cookies** set to **None** and **Query strings** to **All**.
- Click **Save changes**.
- Click **Create** to add a last cache behavior for the MediaPackage origin.
- Enter "/out/v1/*" for the **Path pattern**.
- Under **Origin**, select the MediaPackage origin (**EMP-***).
- Set the **Cache key and origin requests** to **Legacy cache settings** with **Headers**/**Cookies** set to **None** and **Query strings** to **All**.
- Click **Save changes**.

- Remove the behavior with the **Path pattern** set to “*”.
- Under the **Behaviors** tab, double-check the precedence of the caching behaviors because these matters. Of primary precedence should be the MediaTailor origin (precedence 0), followed by the MediaPackage origin (precedence 1), followed lastly by the ad’s origin (precedence 2 by default). If this precedence is not reflected, select one of the behaviors and adjust the precedence by clicking on either **Move up** or **Move down**.

NOTE: When you make changes to the Amazon CloudFront distribution, it will go into an *In progress* state. Once it’s back to the *Deployed* state, your changes will have taken effect.



Step 5: Test your playback URLs

Now that you have completed the deployment and once your MediaLive channel has started, you should be able to play back your content in HLS and MPEG-DASH from the Amazon CloudFront distribution.

- To validate the output of MediaTailor, you can construct your playback URL from the EMP endpoints using the following format:
    - HLS: https://hostname-CloudFront/v1/master/hashed-account-id/origin-id/out/v1/hashed-id-hls/master.m3u8
    - MPEG-DASH: https://hostname-CloudFront /v1/master/hashed-account-id/origin-id/out/v1/hashed-id-dash/manifest.mpd

Step 6: Cleanup

If you followed these instructions, make sure that you delete the AWS resources after your testing is completed to avoid incurring any unnecessary cost in your AWS account.

- In MediaConnect, stop the flow; then delete it.
- In MediaLive, stop the channel; then delete it.
- In MediaPackage, delete all the endpoints; then delete the channel.
- In MediaTailor, delete the configuration.
- In Amazon CloudFront, delete the distribution.