AWS Elemental MediaPackage: AWS Elemental MediaPackage User Guide

Copyright © 2023 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.
# Table of Contents

What is AWS Elemental MediaPackage? ................................................................. 1
Are you a first-time user of MediaPackage? ...................................................... 1
Concepts and terminology .................................................................................. 2
  Live components ............................................................................................. 2
Supported inputs and outputs ............................................................................. 3
  Live supported codecs and input types ......................................................... 3
How MediaPackage works .................................................................................. 4
  Live content processing .................................................................................. 4
Features of AWS Elemental MediaPackage ..................................................... 7
Related services ................................................................................................. 8
Accessing MediaPackage .................................................................................. 8
Pricing for MediaPackage .................................................................................. 9
Regions for MediaPackage ............................................................................... 9
Setting up MediaPackage .................................................................................. 10
  Preliminary steps for setting up ................................................................. 10
    Sign up for an AWS account .................................................................... 10
    Create an administrative user ................................................................. 10
    Download tools ......................................................................................... 11
  Setting up IAM permissions .......................................................................... 11
    Create a role in the IAM console ............................................................. 12
    Assume the role from the IAM console or AWS CLI .............................. 13
    Resource Groups—tagging ....................................................................... 13
Getting started ................................................................................................. 15
Live content delivery .......................................................................................... 15
  Prerequisites ................................................................................................. 15
    Step 1: Access MediaPackage .................................................................. 15
    Step 2: Create a channel group .............................................................. 15
    Step 3: Create a channel ......................................................................... 16
    Step 4: Create an endpoint ...................................................................... 16
    Step 5: Clean up ......................................................................................... 16
Access control .................................................................................................. 18
  Creating new resources ................................................................................ 18
  Sharing resources ......................................................................................... 18
  Protecting data ............................................................................................. 18
Delivering live content ...................................................................................... 20
  Working with channel groups .................................................................... 20
    Creating a channel group ........................................................................ 20
    Viewing channel group details .............................................................. 21
    Editing a channel group ......................................................................... 22
    Deleting a channel group ....................................................................... 22
    Adding channels to a channel group ....................................................... 22
  Working with channels ............................................................................... 23
    Creating a channel .................................................................................. 23
    Viewing channel details ......................................................................... 24
    Editing a channel ..................................................................................... 24
    Deleting a channel .................................................................................. 25
    Adding origin endpoints to a channel ..................................................... 25
  Working with endpoints ............................................................................... 25
    Creating an origin endpoint ................................................................... 26
    Viewing an origin endpoint .................................................................... 31
    Editing an endpoint ................................................................................ 31
    Deleting an endpoint .............................................................................. 32
    Previewing a manifest .......................................................................... 32
Delivering VOD content .................................................................................... 33
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MediaPackage features</td>
<td>34</td>
</tr>
<tr>
<td>Content encryption and DRM</td>
<td>34</td>
</tr>
<tr>
<td>Limitations and requirements</td>
<td>34</td>
</tr>
<tr>
<td>Container and DRM system support with SPEKE</td>
<td>35</td>
</tr>
<tr>
<td>Deploying SPEKE</td>
<td>35</td>
</tr>
<tr>
<td>Understanding key rotation behavior</td>
<td>36</td>
</tr>
<tr>
<td>SPEKE Version 2.0 presets</td>
<td>36</td>
</tr>
<tr>
<td>Manifest filtering</td>
<td>38</td>
</tr>
<tr>
<td>Working with manifest filters</td>
<td>39</td>
</tr>
<tr>
<td>Manifest filter query parameters</td>
<td>40</td>
</tr>
<tr>
<td>Manifest filtering examples</td>
<td>43</td>
</tr>
<tr>
<td>Special conditions for TS and CMAF manifests</td>
<td>43</td>
</tr>
<tr>
<td>Error conditions</td>
<td>44</td>
</tr>
<tr>
<td>Metadata passthrough</td>
<td>45</td>
</tr>
<tr>
<td>ID3 metadata considerations</td>
<td>45</td>
</tr>
<tr>
<td>Rendition groups</td>
<td>45</td>
</tr>
<tr>
<td>When to use rendition groups</td>
<td>46</td>
</tr>
<tr>
<td>When not to use rendition groups</td>
<td>46</td>
</tr>
<tr>
<td>SCTE-35 messages</td>
<td>46</td>
</tr>
<tr>
<td>SCTE-35 settings in MediaPackage</td>
<td>47</td>
</tr>
<tr>
<td>How it works</td>
<td>48</td>
</tr>
<tr>
<td>EXT-X-DATERANGE ad markers</td>
<td>48</td>
</tr>
<tr>
<td>Time-shifted viewing</td>
<td>50</td>
</tr>
<tr>
<td>Time delay</td>
<td>51</td>
</tr>
<tr>
<td>Rules for start and end parameters</td>
<td>51</td>
</tr>
<tr>
<td>Trick-play</td>
<td>52</td>
</tr>
<tr>
<td>Using I-frame playlists to enable trick-play</td>
<td>52</td>
</tr>
<tr>
<td>Data plane APIs</td>
<td>53</td>
</tr>
<tr>
<td>PutObject</td>
<td>53</td>
</tr>
<tr>
<td>GetObject</td>
<td>53</td>
</tr>
<tr>
<td>GetHeadObject</td>
<td>54</td>
</tr>
<tr>
<td>Security</td>
<td>55</td>
</tr>
<tr>
<td>Data protection</td>
<td>55</td>
</tr>
<tr>
<td>Implementing DRM</td>
<td>56</td>
</tr>
<tr>
<td>Identity and Access Management</td>
<td>56</td>
</tr>
<tr>
<td>Audience</td>
<td>57</td>
</tr>
<tr>
<td>Authenticating with identities</td>
<td>57</td>
</tr>
<tr>
<td>Managing access using policies</td>
<td>59</td>
</tr>
<tr>
<td>How AWS Elemental MediaPackage works with IAM</td>
<td>61</td>
</tr>
<tr>
<td>Identity-based policy examples</td>
<td>66</td>
</tr>
<tr>
<td>Resource-based policy examples</td>
<td>68</td>
</tr>
<tr>
<td>AWS managed policies</td>
<td>75</td>
</tr>
<tr>
<td>Authenticating Requests</td>
<td>77</td>
</tr>
<tr>
<td>Cross-service confused deputy prevention</td>
<td>77</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>78</td>
</tr>
<tr>
<td>Learn More</td>
<td>79</td>
</tr>
<tr>
<td>Compliance validation</td>
<td>80</td>
</tr>
<tr>
<td>Resilience</td>
<td>80</td>
</tr>
<tr>
<td>Infrastructure Security</td>
<td>80</td>
</tr>
<tr>
<td>Logging and monitoring</td>
<td>82</td>
</tr>
<tr>
<td>Monitoring with CloudWatch metrics</td>
<td>82</td>
</tr>
<tr>
<td>Live content metrics</td>
<td>83</td>
</tr>
<tr>
<td>Logging AWS Elemental MediaPackage API calls with AWS CloudTrail</td>
<td>91</td>
</tr>
<tr>
<td>MediaPackage information in CloudTrail</td>
<td>92</td>
</tr>
<tr>
<td>Understanding MediaPackage log file entries</td>
<td>92</td>
</tr>
<tr>
<td>Monitoring manifest update time</td>
<td>93</td>
</tr>
<tr>
<td>X-MediaPackage-Manifest-Last-Sequence</td>
<td>93</td>
</tr>
</tbody>
</table>
What is AWS Elemental MediaPackage?

Note
This user guide is intended for creating MediaPackage resources in MediaPackage Version 2 (v2) starting from May 2023. To get started with MediaPackage v2, create your MediaPackage resources. There isn't an automated process to migrate your resources from MediaPackage v1 to MediaPackage v2.
The names of the entities that you use to access your MediaPackage resources, like URLs and ARNs, all include "mediapackagev2", to distinguish from the prior version. If you used MediaPackage prior to this release, you can't use the MediaPackage v2 AWS CLI or the MediaPackage v2 API to access any MediaPackage v1 resources.
If you created resources in MediaPackage v1, use video on demand (VOD) workflows, and aren't looking to migrate to MediaPackage v2 yet, see the AWS Elemental MediaPackage v1 User Guide.

AWS Elemental MediaPackage (MediaPackage) is a just-in-time video packaging and origination service that runs in the AWS Cloud. With MediaPackage, you can deliver highly secure, scalable, and reliable video streams to a wide variety of playback devices and content delivery networks (CDNs).

MediaPackage offers a broadcast-grade viewing experience for viewers, while allowing you the flexibility to control and protect your content. Additionally, the built-in resiliency and scalability of MediaPackage means that you have the right amount of resources at the right time, with no manual intervention required.

Topics
• Are you a first-time user of MediaPackage? (p. 1)
• Concepts and terminology (p. 2)
• Supported inputs and outputs (p. 3)
• How MediaPackage works (p. 4)
• Features of AWS Elemental MediaPackage (p. 7)
• Related services (p. 8)
• Accessing MediaPackage (p. 8)
• Pricing for MediaPackage (p. 9)
• Regions for MediaPackage (p. 9)

Are you a first-time user of MediaPackage?

If you’re a first-time user of MediaPackage, we recommend that you begin by reading the following sections:
• Concepts and terminology (p. 2)
• How MediaPackage works (p. 4)
• Features of AWS Elemental MediaPackage (p. 7)
Concepts and terminology

AWS Elemental MediaPackage includes the following components:

**Just-in-time packaging**

MediaPackage performs *just-in-time packaging* (JITP). When a playback device requests content, MediaPackage dynamically customizes the live video streams and creates a manifest in a format that's compatible with the requesting device.

**Origination service**

MediaPackage is considered an *origination service* because it's the point of distribution for media content delivery.

**Packager**

A *packager* prepares output streams for access by different types of players. The packager type specifies the streaming format that MediaPackage delivers from the endpoint (either Apple HLS, DASH-ISO, Microsoft Smooth Streaming, or Common Media Application Format [CMAF]). Additional packager settings include buffer and update durations and manifest tag handling instructions.

A packager is a part of an origin endpoint. Each endpoint must have one, and only one, packager. To use different packager types for the same content, create multiple endpoints on the channel.

**Source Content**

*Source contents* are live streams and video files that MediaPackage ingests.

- For live video, source content comes from an upstream encoder, such as AWS Elemental MediaLive. MediaPackage supports HLS source content.

**Stream**

A *stream* refers to the content input and output of MediaPackage.

For live workflows, an upstream encoder sends a live stream as an input to MediaPackage to the channel. When a downstream device requests playback of the content, MediaPackage dynamically packages the stream (including specifying the packager type, adding encryption, and configuring track outputs) and delivers it to the requesting device as an output of the endpoint. An endpoint can produce multiple streams.

**Track**

*Tracks* make up the output content stream. MediaPackage includes selected video, audio, and subtitles or captions tracks in the output stream. The stream delivers the tracks to the player (either directly or through a CDN), and the player plays back the tracks based on player logic or network conditions (such as available bandwidth).

**Live components**

The following components apply to live workflows in MediaPackage:

**Channel group**

A *channel group* is the top-level resource that consists of channels and origin endpoints that are associated with it and that provides predictable URLs for stream delivery. All channels and origin endpoints within the channel group are guaranteed to share the DNS.
Channel

A channel represents the entry point for a content stream into MediaPackage. Upstream encoders such as AWS Elemental MediaLive send content to the channel. When MediaPackage receives a content stream, it packages the content and outputs the stream from an endpoint that you create on the channel. There's one channel for each incoming set of adaptive bitrate (ABR) streams.

Endpoint

An endpoint is part of a channel and represents the packaging aspect of MediaPackage. When you create an endpoint on a channel, you indicate what streaming format, packaging parameters, and features the output stream will use. Downstream devices request content from the endpoint. A channel can have multiple endpoints.

Supported inputs and outputs

This section describes the input types, input codecs, and output codecs that AWS Elemental MediaPackage supports for live content.

Topics

• Live supported codecs and input types (p. 3)

Live supported codecs and input types

The following sections describe supported input types and codecs for live streaming content.

Supported input types

These are the input types that MediaPackage supports for live content.

<table>
<thead>
<tr>
<th>MediaPackage input type</th>
<th>Use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLS</td>
<td>Push an HLS stream from an external source or encoder (such as AWS Elemental MediaLive) using the HTTPS protocol. Additional requirements:</td>
</tr>
<tr>
<td></td>
<td>• You must define a channel policy to enable content to flow into your channel from sources outside of your account.</td>
</tr>
<tr>
<td></td>
<td>• Media segments must not be encrypted.</td>
</tr>
<tr>
<td></td>
<td>• Streams can contain either muxed video and audio tracks, or unmuxed tracks.</td>
</tr>
<tr>
<td></td>
<td>• The input must contain at least one video track. MediaPackage doesn't support inputs that contain no video track.</td>
</tr>
</tbody>
</table>

Supported input codecs

These are the video, audio, and subtitles codecs that MediaPackage supports for source content streams.
# How MediaPackage works

AWS Elemental MediaPackage uses just-in-time format conversion to deliver over-the-top (OTT) video from a single source to a wide variety of playback devices or content delivery networks (CDNs).

## Live content processing

In the processing flow for live content, encoders send live HLS streams to MediaPackage. MediaPackage then packages the content, formatting it in response to playback requests from downstream devices.

The following sections describe the live processing flows.

### Topics
- General MediaPackage live processing flow (p. 4)
- Live input redundancy AWS Elemental MediaPackage processing flow (p. 6)

## General MediaPackage live processing flow

The following outlines the general flow of live content in MediaPackage:

### Supported output codecs

These are the video, audio, and subtitles codecs that MediaPackage supports when delivering live content.

<table>
<thead>
<tr>
<th>Endpoint type</th>
<th>Manifest format</th>
<th>Media container</th>
<th>Video codecs</th>
<th>Audio codecs</th>
<th>Subtitles/captions format</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS</td>
<td>HLS</td>
<td>Video: TS</td>
<td>H.264 (AVC)</td>
<td>AAC</td>
<td>WebVTT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Audio: TS or AAC</td>
<td>H.265 (HEVC) with HDR-10 support</td>
<td>Dolby Digital</td>
<td>CEA-608 and CEA-708 closed captions</td>
</tr>
<tr>
<td>CMAF</td>
<td>HLS</td>
<td>CMAF</td>
<td>H.264 (AVC)</td>
<td>AAC</td>
<td>WebVTT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>H.265 (HEVC) with HDR-10 support</td>
<td>Dolby Digital</td>
<td>CEA-608 and CEA-708 closed captions</td>
</tr>
</tbody>
</table>

### How MediaPackage works

AWS Elemental MediaPackage uses just-in-time format conversion to deliver over-the-top (OTT) video from a single source to a wide variety of playback devices or content delivery networks (CDNs).
1. An upstream encoder (such as AWS Elemental MediaLive) sends an HLS live stream using AWS Signature Version 4 to authorize request to your origin and your IAM channel policy. If you’re using input redundancy, the encoder sends two identical HLS live streams to MediaPackage, one to each ingest domain on the channel. MediaPackage uses the stream from one ingest URL as the source content. If MediaPackage stops receiving content on the active ingest URL, it automatically switches to the other ingest URL for source content. Additionally, AWS scales resources up and down to handle the incoming traffic.

For more information, see Live input redundancy AWS Elemental MediaPackage processing flow (p. 6).

Note
To permit support for features like time-shifted viewing, MediaPackage stores all received content for a limited time. This stored content is only available for playback if it falls within the startover window that's defined on the endpoint. Stored content isn't available for playback if it's outside the startover window, or if you haven't defined a window on the endpoint. For more information, see Time-shifted viewing reference in AWS Elemental MediaPackage (p. 50).

2. A downstream device requests content from MediaPackage through the endpoint egress domain. A downstream device is either a video player or a CDN. The egress domain is associated with a channel group and an endpoint for a specific streaming format (either TS or CMAF).

3. When MediaPackage receives the playback request from the downstream device, it dynamically packages the stream according to the settings that you specified on the origin endpoint. Packaging can include adding encryption and configuring audio, video, and subtitles or captions track outputs.

4. MediaPackage delivers the output stream over HTTPS to the requesting device. As with input, AWS scales resources up and down to handle changes in traffic.

Throughout the content input and output processes, MediaPackage detects and mitigates potential infrastructure failures before they become a problem for viewers.

The following illustration shows the overall process.
Live input redundancy AWS Elemental MediaPackage processing flow

Achieve input redundancy in AWS Elemental MediaPackage by sending two streams to separate ingest domains on a channel in MediaPackage. One of the streams becomes the primary, active source of content for the endpoints, while the other continues to passively receive content. If MediaPackage stops receiving content from the active stream, it switches over to the other ingest stream so that content playback isn't interrupted.

If you use MediaPackage with AWS Elemental MediaLive (for example), here's the flow of input redundancy:

1. You create a channel group in MediaPackage, as described in Creating a channel group (p. 20). When MediaPackage provisions the channel group, it creates an egress domain for all channels and origin endpoints within the channel group.
2. You create a channel within the channel group as described in Creating a channel (p. 23). When MediaPackage provisions the channel, it creates two ingest domains for the channel. If you're not using input redundancy, you can send a stream to either ingest domain. There's no requirement that you send content to both domains.
3. You create an origin endpoint within the channel as described in Creating an origin endpoint (p. 26).

   **Important**
   If you use short output segments, depending on your playback device, you might see buffering when MediaPackage switches inputs. You can reduce buffering by using the time delay feature on the endpoint. Be aware that using a time delay introduces latency to end-to-end delivery of the content. For information about enabling time delay, see Creating an origin endpoint (p. 26).

4. You create an input and channel in AWS Elemental MediaLive, and you add a MediaPackage output group to the channel in MediaLive. For more information, see Creating a Channel from Scratch in the AWS Elemental MediaLive User Guide.

   If you use an HLS output group in AWS Elemental MediaLive, the input loss action on the HLS group's settings must be set to pause the output if the service doesn't receive input. If MediaLive sends a black frame or some other filler frame when it's missing input, then MediaPackage can't tell when segments are missing, and subsequently can't perform failover. For more information about setting the input loss action in MediaLive, see Fields for the HLS Group in the AWS Elemental MediaLive User Guide.

   **Important**
   If you use a different encoder (not AWS Elemental MediaLive) and you send two separate streams to the same channel in MediaPackage, the streams must have identical encoder settings and manifest names. Otherwise, input redundancy might not work correctly and playback could be interrupted if the inputs switch.

5. You start the channel in AWS Elemental MediaLive to send the streams to MediaPackage.

6. MediaPackage receives content on both of the ingest URLs, but only one of the streams is used for source content at a time. If the active stream is missing any segments, then MediaPackage automatically fails over to the other stream. MediaPackage continues to use this stream until failover is needed again.

   The formula that's used to determine if an input is missing segments is based on the segment lengths on the inputs and the endpoints. If an input is missing segments and quickly recovers, an endpoint with longer segment lengths won't switch inputs. This might result in different endpoints on the channel using different inputs (if one endpoint switches and the other doesn't). This is expected behavior and should not affect the content workflow.
Features of AWS Elemental MediaPackage

MediaPackage supports the following features:

Audio

MediaPackage supports multi-language audio inputs and the following audio codecs:

• AAC stereo
• Dolby AC3 and E-AC3 (Dolby Digital and Dolby Digital+)

MediaPackage accepts these codecs from the input source and passes them through to the output stream.

Important
MediaPackage doesn't support audio-only inputs. The stream configuration from the encoder must include at least one video track.

Captions

Your embedded source captions can be CEA-608 captions, CEA-708 captions, or both CEA-608 and CEA-708. MediaPackage will pass through these captions in the media segments on TS and CMAF origin endpoints, and generate the appropriate manifest signaling.

Important
Your input HLS playlist must include captions signaling tags. If not present, MediaPackage will not be able to generate the corresponding output manifest signaling.

DRM

MediaPackage supports content protection through digital rights management (DRM). For information, see Content encryption and DRM in AWS Elemental MediaPackage (p. 34).

HLS Rendition Groups

MediaPackage supports rendition groups for incoming and outgoing HLS content. For information about output rendition groups, see Rendition groups reference in AWS Elemental MediaPackage (p. 45).

Input Redundancy

Input redundancy is available with only live workflows in MediaPackage.

MediaPackage creates two ingest URLs on every channel group so that you can create input redundancy by sending two identical streams to the same channel. For information about how input redundancy works, see Live input redundancy AWS Elemental MediaPackage processing flow (p. 6).

Low-latency streaming

MediaPackage supports Apple low-latency HLS, which is a technology aimed at reducing the delay between the time content is captured and the time it is displayed on the viewer's screen. The goal is to achieve minimal end-to-end delay (or "glass-to-glass" delay) by using techniques such as parallel delivery and reduced buffering. This technology enables a more seamless and immersive real-time viewing experience for users, particularly in applications such as live video streaming, teleconferencing, and online gaming.

Subtitles

MediaPackage supports input WebVTT text-based subtitles. MediaPackage translates the subtitles to the appropriate format based on the packager that's used on the endpoint:

• For TS and CMAF: WebVTT is passed through
Time-shift Viewing

Time-shift viewing is available with only live workflows in MediaPackage.

MediaPackage supports playback of a stream at a time earlier than the current time. Start-over, catch-up TV, and time delay are all supported. For more information about setting up time-shift capabilities, see Time-shifted viewing reference in AWS Elemental MediaPackage (p. 50).

Video

MediaPackage supports the input H.264 video codec and passes it through to the output stream. CMAF endpoints in MediaPackage also support H.265/HEVC and HDR-10, following the Apple specification to applicable playback devices.

Important

MediaPackage requires at least one video track to be present in the stream configuration from the encoder. The service doesn't support audio-only ingest.

Related services

- **Amazon CloudWatch** is a monitoring service for AWS Cloud resources and the applications that you run on AWS. Use CloudWatch to track metrics such as content input and output request counts. For more information, see Amazon CloudWatch.
- **AWS Elemental MediaLive (MediaLive)** is a live video processing service that encodes high-quality live video streams for broadcast television and multi-screen devices. Use MediaLive to encode content streams and send them to MediaPackage for packaging. For more information about how encoders (such as MediaLive) work with MediaPackage, see How MediaPackage works (p. 4).
- **AWS Elemental MediaTailor (MediaTailor)** is a scalable ad insertion service that runs in the AWS Cloud. Use MediaTailor to serve targeted ads to viewers. For more information, see AWS Elemental MediaTailor.
- **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 users can use in which ways (authorization). For more information, see the section called “Preliminary steps for setting up” (p. 10).

Accessing MediaPackage

You can access MediaPackage using any of the following methods.

- **AWS Management Console** - The procedures throughout this guide explain how to use the AWS Management Console to perform tasks for MediaPackage.

  https://console.aws.amazon.com/mediapackage/

- **AWS Command Line Interface** - For more information, see the AWS Command Line Interface User Guide.

  aws mediapackagev2

- **MediaPackage API** - For information about API actions and about how to make API requests, see the AWS Elemental MediaConnect API Reference.

  https://mediapackagev2.region.amazonaws.com
• **AWS SDKs** - If you're using a programming language that AWS provides an SDK for, you can use an SDK to access MediaPackage. SDKs simplify authentication, integrate easily with your development environment, and provide easy access to MediaPackage commands. For more information, see [Tools for Amazon Web Services](https://aws.amazon.com/documentation/tools/).


### Pricing for MediaPackage

As with other AWS products, there are no contracts or minimum commitments for using MediaPackage. You're charged only for AWS resources that your account uses. Pricing is pay-as-you-go and consists of the following:

- A per GB charge for received content
- A per GB charge for content that's streamed out of MediaPackage

Content that's cached and served from a content delivery network (CDN) doesn't incur this per GB charge.

For detailed pricing information, see [MediaPackage Pricing](https://aws.amazon.com/media-package/pricing/).

### Regions for MediaPackage

To reduce latency in your applications, MediaPackage offers a regional endpoint for your requests. To view the list of AWS Regions where MediaPackage is available, see [MediaPackage Regions](https://aws.amazon.com/media-package/regions/).
Setting up MediaPackage

This section provides procedures to set up your organization to use AWS Elemental MediaPackage. It also provides information about determining the IAM permissions that users and other AWS identities require. These permissions let you impose restricted controls on users and other AWS identities, in conformance with the security policies and procedures of your organization.

Topics
- Preliminary steps for setting up (p. 10)
- Setting up IAM permissions (p. 11)

Preliminary steps for setting up

This topic describes preliminary steps, such as creating an account, to prepare you to use MediaPackage. You aren't charged for these preliminary items. You are charged only for AWS services that you use.

Topics
- Sign up for an AWS account (p. 10)
- Create an administrative user (p. 10)
- Download tools (p. 11)

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

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 an administrative 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 an administrative user

After you sign up for an AWS account, 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 an administrative user

- For your daily administrative tasks, grant administrative access to an administrative user in AWS IAM Identity Center.

   For instructions, see Getting started in the AWS IAM Identity Center User Guide.

Sign in as the administrative user

- 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.

Download tools

The AWS Management Console includes a console for MediaPackage, but if you want to access the services programmatically, see the following:

- The API guides document the operations that the services support and provide links to the related SDK and CLI documentation:
  - AWS Elemental MediaPackage API Reference
- To call an API without having to handle low-level details like assembling raw HTTP requests, you can use an AWS SDK. The AWS SDKs provide functions and data types that encapsulate the functionality of AWS services. To download an AWS SDK and access installation instructions, see the applicable page:
  - Go
  - JavaScript
  - .NET
  - Node.js
  - Python
  - Ruby

For a complete list of AWS SDKs, see Tools for Amazon Web Services.

- You can use the AWS Command Line Interface (AWS CLI) to control multiple AWS services from the command line. You can also automate your commands using scripts. For more information, see AWS Command Line Interface.

- AWS Tools for Windows PowerShell supports these AWS services. For more information, see AWS Tools for PowerShell Cmdlet Reference.

Setting up IAM permissions

By default, users and roles don't have permission to create or modify MediaPackage 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 Creating IAM policies in the IAM User Guide.

For details about actions and resource types defined by MediaPackage, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Elemental MediaPackage in the Service Authorization Reference.

This section describes the permissions that you must assign to users and other AWS identities so that they can work with MediaPackage and other AWS services that your workflows use. After you have identified the required permissions, you will be able to design and create the relevant policies, and attach those policies to groups of users or to roles.

This section assumes that you have already performed these tasks:

- You have performed the initial setup described in Preliminary steps for setting up (p. 10) in order to sign up for MediaPackage and to create an administrator.
- You have read the recommendations in Identity and Access Management for AWS Elemental MediaPackage (p. 56) about how to create administrators, users, and other AWS identities.

Topics

- Create a role in the IAM console (p. 12)
- Assume the role from the IAM console or AWS CLI (p. 13)
- Requirements for AWS Resource Groups—tagging (p. 13)

Create a role in the IAM console

Create a role in the IAM console for each policy that you create. This allows users to assume a role rather than attaching individual policies to each user.

To create a role in the IAM console

1. Sign in to the AWS Management Console and 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. Under Select trusted entity, choose AWS account.
4. Under An AWS account, select the account with the users that will be assuming this role.
   - If a third-party will be accessing this role, it's best practice to select Require external ID. For more information about external IDs, see Using an external ID for third-party access in the IAM User Guide.
   - It's best practice to require multi-factor authentication (MFA). You can select the check box next to Require MFA. For more information about MFA, see Multi-factor authentication (MFA) in the IAM User Guide.
5. Choose Next.
6. Under Permissions policies, search for and add the policy with the appropriate MediaPackage permissions level.
   - For access to live functionality, choose one of the following options:
     - Use AWSElementalMediaPackageFullAccess to allow the user to perform all actions on all live resources in MediaPackage.
7. Add policies to allow the MediaPackage console to make calls to Amazon CloudWatch on the user's behalf. Without these policies, the user is able to use the service's API only (not the console). Choose one of the following options:

- Use `AWSElementalMediaPackageReadOnly` to provide the user read-only rights for all live resources in MediaPackage.

- Use `ReadOnlyAccess` to allow MediaPackage to communicate with CloudWatch, and also provide the user read-only access to all AWS services on your account.

- Use `CloudWatchReadOnlyAccess`, `CloudWatchEventsReadOnlyAccess`, and `CloudWatchLogsReadOnlyAccess` to allow MediaPackage to communicate with CloudWatch, and limit the user's read-only access to CloudWatch.

8. (Optional) Set a permissions boundary. This is an advanced feature that is available for service roles, but not service-linked roles.

   1. Expand the Permissions boundary section and choose Use a permissions boundary to control the maximum role permissions. IAM includes a list of the AWS managed and customer managed policies in your account.
   2. Select the policy to use for the permissions boundary or choose Create policy to open a new browser tab and create a new policy from scratch. For more information, see Creating IAM policies in the IAM User Guide.
   3. After you create the policy, close that tab and return to your original tab to select the policy to use for the permissions boundary.

9. Verify that the correct policies are added to this group, and then choose Next.

10. If possible, enter a role name or role name suffix to help you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot create roles named both `PRODROLE` and `prodrole`. Because various entities might reference the role, you cannot edit the name of the role after it has been created.

11. (Optional) For Description, enter a description for the new role.

12. Choose Edit in the Step 1: Select trusted entities or Step 2: Select permissions sections to edit the use cases and permissions for the role.

13. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see Tagging IAM resources in the IAM User Guide.

14. Review the role and then choose Create role.

Assume the role from the IAM console or AWS CLI

View the following resources for learning about granting permissions for users to assume the role and how users can switch to the role from the IAM console or AWS CLI.

- For more information about granting a user permissions to switch roles, see Granting a user permissions to switch roles in the IAM User Guide.

- For more information about switching roles (console), see Switching to a role (console) in the IAM User Guide.

- For more information about switching roles (AWS CLI), see Switching to an IAM role (AWS CLI) in the IAM User Guide.

Requirements for AWS Resource Groups—tagging

When users create channel groups, channels, or origin endpoints, they can optionally attach tags to the resource during creation. Typically, your organization has a policy to tag or to omit tags. There are two services that control permissions for tagging, for two different scenarios:
The ability to tag during channel creation is controlled by actions within MediaPackage.
The ability to modify tags in existing resources is controlled by actions within Resource Group Tagging. See Working with Tag Editor in Getting Started with the AWS Management Console.
Getting started with AWS Elemental MediaPackage

The following sections describe how to quickly get started receiving and sending content with AWS Elemental MediaPackage.

Topics

- Getting started with live content delivery in AWS Elemental MediaPackage (p. 15)
- Access control best practices (p. 18)

Getting started with live content delivery in AWS Elemental MediaPackage

This Getting Started tutorial shows you how to use the MediaPackage console to create a channel and endpoints for streaming live videos.

Topics

- Prerequisites (p. 15)
- Step 1: Access MediaPackage (p. 15)
- Step 2: Create a channel group (p. 15)
- Step 3: Create a channel (p. 16)
- Step 4: Create an endpoint (p. 16)
- Step 5: Clean up (p. 16)

Prerequisites

Before you can use MediaPackage, you need an AWS account and the appropriate permissions to access, view, and edit MediaPackage components. Make sure that your system administrator has completed the steps in Setting up MediaPackage (p. 10), and then return to this tutorial.

For supported live inputs and codecs, see Live supported codecs and input types (p. 3).

Step 1: Access MediaPackage

Using your IAM credentials, sign in to the AWS Elemental MediaPackage console:

https://console.aws.amazon.com/mediapackage/

Step 2: Create a channel group

A channel group is the top-level resource that streamlines the organization of multiple channels and origin endpoints associated with it.
To create a channel group

1. Open the MediaPackage console at https://console.aws.amazon.com/mediapackage/.
2. On the Channel groups page, choose Create channel group.
3. Enter a unique name that describes the channel group and, optionally, a description.
4. Choose Create.
   
   MediaPackage displays the new channel group's details page.

Step 3: Create a channel

The channel represents the input to MediaPackage for incoming live content from an encoder such as AWS Elemental MediaLive. The channel receives content, and after packaging it, outputs it through an endpoint to downstream devices (such as video players or CDNs) that request the content.

MediaPackage does not require that you supply any customer data. There are no fields in channels where there is an expectation that you will provide customer data.

To create a channel

1. Access the channel group that the channel will be associated with.
2. In the Channel group details page, under Channels, choose Create channel.
3. Enter a unique name that describes the channel and, optionally, a description.
4. Choose your channel's IAM policy that defines the permissions of your channel.
5. Choose Create.
   
   MediaPackage displays the new channel's details page. The channel is active and can start receiving content as soon as it's created.

Step 4: Create an endpoint

The endpoint is attached to a channel, and represents the output of the live content. You can associate multiple endpoints to a single channel. Each endpoint gives players and downstream CDNs (such as Amazon CloudFront) access to the content for playback.

To create an endpoint

1. On the Channels page, choose the channel that the endpoint will be associated with.
2. On the details page for the channel, under Origin endpoints, choose Create endpoint.
3. Enter a unique name that describes the endpoint and, optionally, a description.
4. Choose the container type and define the corresponding settings.
5. Choose your origin endpoint's IAM policy that defines the permissions of your endpoint.
6. Define the manifests emitted from the origin endpoint.
7. Choose Save.
   
   MediaPackage displays the channel's details page, including the endpoint that you just created.

Step 5: Clean up

To avoid extraneous charges, be sure to delete all unnecessary channel groups, channels, and endpoints. You must delete the channels and endpoints before you can delete the channel group.
1. Delete the endpoint as described in Deleting an endpoint (p. 32).
2. Delete a channel as described in Deleting a channel (p. 25).
3. Delete the channel group as described in Deleting a channel group (p. 22).
Access control best practices

MediaPackage provides a variety of security features and tools. The following scenarios should serve as a guide to what tools and settings you might want to use when performing certain tasks or operating in specific environments. Proper application of these tools can help maintain the integrity of your data and help ensure that your resources are accessible to the intended users.

Topics

• Creating new resources (p. 18)
• Sharing resources (p. 18)
• Protecting data (p. 18)

Creating new resources

When creating new resources, you should apply the following tools and settings to help ensure that your MediaPackage resources are protected.

Grant access with IAM identities

When setting up accounts for new team members who require MediaPackage access, use IAM users and roles to ensure least privileges. You can also implement a form of IAM multi-factor authentication (MFA) to support a strong identity foundation. Using IAM identities, you can grant unique permissions to users and specify what resources they can access and what actions they can take. IAM identities provide increased capabilities, including the ability to require users to enter login credentials before accessing shared resources and apply permission hierarchies to different objects within a single bucket.

For more information, see Identity and Access Management for AWS Elemental MediaPackage (p. 56).

Resource policies

With resource policies, you can personalize channel and origin endpoint access to help ensure that only those users you have approved can access resources and perform actions within them. In addition to resource policies, you should use resource-level Block Public Access settings to further limit public access to your data.

For more information, see Resource-based policy examples (p. 68).

When creating policies, avoid the use of wildcard characters in the Principal element because it effectively allows anyone to access your MediaPackage resources. It's better to explicitly list users or groups that are allowed to access the resource. Rather than including a wildcard for their actions, grant them specific permissions when applicable.

To further maintain the practice of least privileges, Deny statements in the Effect element should be as broad as possible and Allow statements should be as narrow as possible. Deny effects paired with the "mediapackagev2:*" action are another good way to implement opt-in best practices for the users included in policy condition statements.

Sharing resources

There are several different ways that you can share resources with a specific group of users. You can use the following tools to share a set of documents or other resources to a single group of users, department, or an office. Although they can all be used to accomplish the same goal, some tools might pair better than others with your existing settings.

User policies
You can share resources with a limited group of people using IAM groups and user policies. When creating a new IAM user, you are prompted to create and add them to a group. However, you can create and add users to groups at any point. If the individuals you intend to share these resources with are already set up within IAM, you can add them to a common group and share the bucket with their group within the user policy. You can also use IAM user policies to share individual objects within a bucket.

For more information, see Identity-based policy examples for MediaPackage (p. 66).

Tagging

If you use object tagging to categorize storage, you can share objects that have been tagged with a specific value with specified users. Resource tagging allows you to control access to objects based on the tags associated with the resource that a user is trying to access. To do this, use the ResourceTag/key-name condition within an IAM user policy to allow access to the tagged resources.

For more information, see Controlling access to AWS resources using resource tags in the IAM User Guide.

Protecting data

Use the following tools to help protect data in transit and at rest, both of which are crucial in maintaining the integrity and accessibility of your data.

Signing methods

AWS Signature Version 4 is the process of adding authentication information to AWS requests sent by HTTP. For security, most requests to AWS must be signed with an access key, which consists of an access key ID and secret access key. These two keys are commonly referred to as your security credentials. For more information, see Authenticating Requests (AWS Signature Version 4) (p. 77) and Signing AWS API requests in the IAM User Guide.
Delivering live content from AWS Elemental MediaPackage

AWS Elemental MediaPackage uses the following resources for live content:

- A **channel group** is the top-level resource that consists of channels and origin endpoints that are associated with it and that provides predictable URLs for stream delivery. All channels and origin endpoints within the channel group are guaranteed to share the DNS.
- A **channel** is the entry point for your live streams from upstream encoders.

  For supported live inputs and codecs, see Live supported codecs and input types (p. 3).
- An **origin endpoint** tells MediaPackage how to package outbound content. Endpoints are associated with channels and hold encryption, stream, and packaging settings.

The following sections describe how to use these resources to manage live content in MediaPackage.

**Topics**

- Working with channel groups in AWS Elemental MediaPackage (p. 20)
- Working with channels in AWS Elemental MediaPackage (p. 23)
- Working with origin endpoints in AWS Elemental MediaPackage (p. 25)

Working with channel groups in AWS Elemental MediaPackage

A channel group is the top-level resource that consists of channels and origin endpoints associated with it. After you create a channel group, MediaPackage provides a fixed egress domain for its lifetime, regardless of any failures or upgrades that might occur. All channels and origin endpoints belonging to this channel group use the same egress domain. Direct your CDNs to this domain for stream delivery from MediaPackage.

For each channel group, you add channels that define the entry point for a content stream into MediaPackage. You then add origin endpoints to the channels that define the packaging options for the output stream.

**Topics**

- Creating a channel group (p. 20)
- Viewing channel group details (p. 21)
- Editing a channel group (p. 22)
- Deleting a channel group (p. 22)
- Adding channels to a channel group (p. 22)

Creating a channel group

This guide shows how to create a channel group as a holder for your channels and origin endpoints. You can provide high-level information about your channel group and can add a certain number of channel groups for each account. After you create a channel group, you can add channels to the channel group.
You can use the MediaPackage console, MediaPackage API, or AWS Command Line Interface (AWS CLI) to create a channel group. When you're creating a channel group, don't put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn't require that you supply any customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

**To create a channel group**

1. Open the MediaPackage console at https://console.aws.amazon.com/mediapackage/.
2. Choose **Create channel group** from the **Channel groups** list.
3. For **Name**, enter a name between 1 through 126 characters that describes the channel group. This is the name that you use for API and console interactions. The name is the primary identifier for the channel group, and must be unique for your account in the AWS Region. Supported characters are A-Z, a-z, 0-9, _ (underscore), and - (hyphen) with a length of 1–256 characters. You can't use spaces in the name, and you can't change the name after you create the channel group.
4. (Optional) For **Description**, enter any descriptive text that helps you to identify the channel group.
5. Choose **Create**.

MediaPackage displays the new channel group's details page.

After you create a channel group, MediaPackage provides an egress domain URL that is fixed for the lifetime of the channel group. This domain remains regardless of any failures or upgrades that might happen over time.

All channels and origin endpoints that belong to this channel group will use the same domain URL. For stream delivery from MediaPackage, direct your CDNs to this domain.

When you create a channel group, if you exceed the quotas on the account, you'll receive an error. The error will be similar to Too many requests, please try again. Resource limit exceeded. This error means that either you exceeded the API request quotas, or that you reached the maximum number of channel groups that your account permits.

**Viewing channel group details**

This guide shows how to view all channel groups that are configured in AWS Elemental MediaPackage. You can also view the details of a specific channel group. This includes the channels and origin endpoints that are associated with it. You can use the MediaPackage console, MediaPackage API, or AWS CLI to view channel group details.

**To view a channel group**

1. Open the MediaPackage console at https://console.aws.amazon.com/mediapackage/.

The console shows all existing channel groups that are configured in MediaPackage.

2. (Optional) To adjust your viewing preferences, choose **Preferences**. For example, you can adjust the page size and properties that you want to view.

3. To view more information about a specific channel group, select that channel group from the **Channel groups** list.

MediaPackage displays important information such as the values for egress domain, when the channel group was created, the ARN, and any channels that are associated with the channel group.
Editing a channel group

This guide shows how to edit the description on a channel group for easier identification later from the AWS Elemental MediaPackage console. You can't edit the name of the channel group.

You can use the MediaPackage console, MediaPackage API, or AWS CLI to edit a channel group. When you're editing a channel group, don't put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn't require that you supply any customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

To edit a channel group

1. Open the MediaPackage console at https://console.aws.amazon.com/mediapackage/.
   The console shows all existing channel groups that are configured in MediaPackage.
2. Select the name of the channel group that you want to edit.
3. On the channel group's details page, choose Edit.
4. Edit the description for easier identification later.
5. Choose Edit.

Deleting a channel group

This guide shows how to delete a channel group to stop AWS Elemental MediaPackage from receiving content. Before you can delete the channel group, you must delete the channel group's channels and endpoints. For instructions, see Deleting a channel (p. 25) and Deleting an endpoint (p. 32). You can use the MediaPackage console, MediaPackage API, or AWS CLI to delete a channel group.

Warning
If you delete a channel group, you'll lose access to the egress domain URL. If that happens, you must create a new channel group to replace it.

To delete a channel group

1. Open the MediaPackage console at https://console.aws.amazon.com/mediapackage/.
   The console shows all existing channel groups that are configured in MediaPackage.
2. Select the name of the channel group that you want to delete.
3. Choose Delete.
4. Choose Delete in the confirmation dialog box.

Adding channels to a channel group

You can add channels to a channel group to do the following:

You can use the AWS Elemental MediaPackage console, MediaPackage API, or the AWS CLI to add channels to a channel group.

For instructions on adding channels to a channel group from the MediaPackage console, see the section called "Working with channels" (p. 23).
Working with channels in AWS Elemental MediaPackage

A channel is part of a channel group and represents the entry point for a content stream into MediaPackage. After you create a channel, MediaPackage provides ingest endpoint domains for its lifetime, regardless of any failures or upgrades that might occur.

Upstream encoders such as AWS Elemental MediaLive send content to the channel. When MediaPackage receives a content stream, it packages the content and outputs the stream from an origin endpoint that you create on the channel. Each incoming set of adaptive bitrate (ABR) streams has one channel. A channel group can have multiple channels.

For supported live inputs and codecs, see [Live supported codecs and input types (p. 3)](#).

Topics
- Creating a channel (p. 23)
- Viewing channel details (p. 24)
- Editing a channel (p. 24)
- Deleting a channel (p. 25)
- Adding origin endpoints to a channel (p. 25)

Creating a channel

This guide shows how to create a channel to start receiving content streams. Later, you add an origin endpoint to the channel. This endpoint is the access point for content playback requests. We recommend that you spread out channels between channel groups, such as putting redundant channels in the same AWS Region in different channel groups.

You can use the MediaPackage console, MediaPackage API, or AWS CLI to create a channel. When you're creating a channel, don't put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn't require that you supply any customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

To create a channel

1. Access the channel group that the channel will be associated with, as described in [Viewing channel group details (p. 21)].
2. Choose Create channel from the Channels list.
3. For Name, enter a name between 1 through 126 characters that describes the channel. The name is the primary identifier for the channel, and must be unique for your account in the AWS Region and channel group. Supported characters are `A-Z`, `a-z`, `0-9`, `_` (underscore), and `-` (hyphen). You can't use spaces in the name, and you can't change the name after you create the channel.

   The name is the primary identifier for the channel, and it must be unique for your account in the AWS Region and channel group.

4. (Optional) For Description, enter descriptive text to help you identify the channel.
5. Choose your channel's IAM policy settings from the following options:
   - **Don't attach a policy** – Restrict access to only those who have access to this account's credentials.

   Choose this option if you're using AWS Elemental MediaLive in the same account as MediaPackage.
- **Attach a custom policy** – Define your own policy and restrict access to as few or as many accounts and resources you want. Enter a valid JSON object with the same structure as other IAM policies. The policy should follow the standard security advice of granting least privilege, or granting only the permissions required to perform a task.

  If you're not using MediaLive, choose this option and define your policy.

  For more information about policies, see [Resource-based policy examples](#) (p. 68).

6. Choose **Create**.

   MediaPackage displays the new channel's details page.

   After you create a channel, MediaPackage provides two ingest endpoint domain URLs that are fixed for the lifetime of the channel. The channel is active and can start receiving content as soon as it's created. Provide this information for the upstream encoder stream destination settings. MediaPackage dynamically adjusts resources to increase capacity for your traffic.

   If you're using input redundancy and one of the inputs stops sending content, then MediaPackage automatically switches to the other input for the source content. For more information about how input redundancy works, see [Live input redundancy](#) (AWS Elemental MediaPackage processing flow) (p. 6).

   While you create a channel, if you exceed the quotas on the account, you'll receive an error. The error will be similar to Too many requests, please try again. Resource limit exceeded. This error means that either you exceeded the API request quotas, or that you reached the maximum number of channels that your account permits.

### Viewing channel details

This guide shows how to view all channels that are configured in AWS Elemental MediaPackage. You can view specific channel details, including the origin endpoints that are associated with it. You can use the MediaPackage console, the AWS Command Line Interface (AWS CLI), or the MediaPackage API to view channel details.

**To view a channel**

1. Access the channel group that the channel is associated with, as described in [Viewing channel group details](#) (p. 21).

   The console shows all existing channels that are configured in MediaPackage.

2. (Optional) To adjust your viewing preferences, choose **Preferences**. For example, you can adjust the page size and properties that you want to view.

3. To view more information about a specific channel, select that channel from the **Channels** list.

   MediaPackage displays important information such as the values for ingest endpoint domain URLs, ARN, and the channel policy.

### Editing a channel

This guide shows how to edit the description on a channel and your channel's policy settings. You can't edit the name of the channel.

You can use the MediaPackage console, MediaPackage API, or AWS CLI to edit a channel. When you're editing a channel, don't put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn't require that you supply any
customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

To edit a channel

1. Access the channel group that the channel is associated with, as described in Viewing channel group details (p. 21).
2. To edit a specific channel, select that channel from the Channels list.
3. On the channel's details page, choose Edit.
4. Make the changes that you want.
5. Choose Update.

Deleting a channel

This guide shows how to delete a channel to stop AWS Elemental MediaPackage from receiving further content. Before you can delete the channel, you must delete the channel's origin endpoints as described in Deleting an endpoint (p. 32). You can use the MediaPackage console, the AWS CLI, or the MediaPackage API to delete a channel.

To delete a channel

1. Access the channel group that the channel is associated with, as described in Viewing channel group details (p. 21).
2. Select the name of the channel that you want to delete.
3. Choose Delete.
4. Choose Delete in the confirmation dialog box.

Adding origin endpoints to a channel

To permit downstream video players and content delivery networks (CDNs) to request content playback, you must add an origin endpoint to a channel.

You can use the AWS Elemental MediaPackage console, MediaPackage API, or the AWS CLI to add origin endpoints to a channel.

For instructions on adding endpoints to a channel from the MediaPackage console, see the section called “Working with endpoints” (p. 25).

Working with origin endpoints in AWS Elemental MediaPackage

An origin endpoint is part of a channel and represents the packaging aspect of MediaPackage. When you create an endpoint on a channel, you indicate what streaming format, packaging parameters, and features the output stream will use. Downstream devices request content from the endpoint. Direct your CDNs to the channel group egress domain for stream delivery from MediaPackage. A channel can have multiple endpoints.

Additionally, the endpoint holds information about digital rights management (DRM) and encryption integration, stream bitrate presentation order, and more.

Topics
Creating an origin endpoint

This guide shows how to create an origin endpoint (endpoint) on a channel to define how MediaPackage prepares content for delivery. Content can't be served from a channel until it has an endpoint. If you're using input redundancy, each endpoint receives content from one ingest URL at a time. If MediaPackage performs a failover on the inputs for one ingest URL, the endpoints automatically start receiving content from the other ingest URL. For more information about input redundancy and failover, see Live input redundancy AWS Elemental MediaPackage processing flow (p. 6).

You can use the MediaPackage console, MediaPackage API, or AWS CLI to create an origin endpoint. When you're creating an origin endpoint, don't put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn’t require that you supply any customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

To create an endpoint

1. Access the channel that the endpoint will be associated with, as described in Viewing channel details (p. 24).
2. Choose Create endpoint from the Origin endpoints list.
3. Complete the fields as described in the following topics:
   - Endpoint settings fields (p. 26)
   - Segment settings fields (p. 27)
   - Encryption fields (p. 27)
   - Endpoint policy fields (p. 29)
   - Manifest fields (p. 29)
4. Choose Create.

When you're creating an endpoint, you will receive an error if you exceed the quotas on the account. An error similar to Too many requests, please try again. Resource limit exceeded means that either you've exceeded the API request quotas, or you've already reached the maximum number of endpoints permitted on this channel.

Endpoint settings fields

The endpoint settings fields hold general information about the endpoint.

1. For Name, enter a name between 1 through 126 characters that describes the origin endpoint. This is the name that you use for API and console interactions. The name is the primary identifier for the endpoint and must be unique for your account in the AWS Region and channel. Supported characters are A-Z, a-z, 0-9, _ (underscore), and - (hyphen) with a length of 1 to 256 characters. You can't use spaces in the name, and you can't change the name after you create the endpoint.
2. (Optional) For Description, enter any descriptive text that helps you to identify the origin endpoint.
3. For Container type, choose the type of container to attach to this origin endpoint. The container type you choose impacts the segment settings, encryption methods, and manifests you can choose.
The container type options are:

- TS
- CMAF

4. For **Startover window (sec.)**, enter the size of the window (in seconds) to create a window of the live stream that's available for on-demand viewing. Viewers can start-over or catch-up on content that falls within the window. The maximum startover window is 1,209,600 seconds (14 days). For more information about implementing start-over and catch-up TV, see [Time-shifted viewing reference in AWS Elemental MediaPackage](p. 50).

**Segment settings fields**

The segment settings fields hold general information about the segment.

1. For **Segment name**, enter a name that describes the segment. The name is the base name of the segment used in all content manifests inside of the endpoint. Supported characters are A-Z, a-z, 0-9, _ (underscore), and - (hyphen) with a length of 1 to 256 characters. You can't use spaces in the name.

2. For **Segment duration (sec.)**, enter the duration (in seconds) of each segment. Enter a value equal to, or a multiple of, the input segment duration. The maximum segment duration is 30 seconds. If the value that you enter is different from the input segment duration, MediaPackage rounds segments to the nearest multiple of the input segment duration.

3. Select **Include IFrame-only stream** to include an additional I-frame only stream along with the other tracks in the manifest. MediaPackage generates an I-frame only stream from the first rendition in the manifest. The service inserts EXT-I-FRAMES-ONLY tags in the output manifest, and then generates and includes an I-frames only playlist in the stream. This playlist enables player functionality like fast forward and rewind.

4. Select **Use audio rendition group** to group all audio tracks into a single rendition group. All other tracks in the stream can be used with any audio rendition from the group. For more information about rendition groups, see [Rendition groups reference in AWS Elemental MediaPackage](p. 45).

5. Select **Include DVB subtitles** to pass through digital video broadcasting (DVB) subtitles into the output. By default, MediaPackage excludes all DVB subtitles from the output.

6. Select **Enable SCTE support** to include SCTE configuration options. If you select this, you can further define your SCTE configuration in additional fields.

7. For **SCTE filtering**, choose the SCTE-35 message types that will be ad markers in the output. If you don't make a selection here, by default, MediaPackage inserts all ad markers in the output manifest.

   - Splice insert
   - Break
   - Provider advertisement
   - Distributor advertisement
   - Provider placement opportunity
   - Distributor placement opportunity
   - Provider overlay placement opportunity
   - Distributor overlay placement opportunity
   - Program

**Encryption fields**

Protect your content from unauthorized use through content encryption and digital rights management (DRM). MediaPackage uses the **AWS Secure Packager and Encoder Key Exchange (SPEKE) API** to facilitate content encryption and decryption by a DRM provider. Using SPEKE, the DRM provider supplies encryption keys to MediaPackage through the SPEKE API. The DRM provider also supplies licenses to
supported media players for decryption. For more information about how SPEKE is used with services and features running in the cloud, see AWS cloud-based architecture in the Secure Packager and Encoder Key Exchange API Specification guide.

**Note**
To encrypt content, you must have a DRM provider, and be set up to use encryption. For information, see the section called “Content encryption and DRM” (p. 34).

1. Choose **Encrypt content** to serve content with copyright protection.
2. For **Encryption method**, choose the encryption method to use. If you don't see your preferred encryption method, confirm you choose the correct container type. The encryption method you choose impacts the DRM system providers you can choose. For supported encryption methods and DRM system providers, see Container and DRM system support with SPEKE (p. 35).
   - The valid encryption methods for TS container types are:
     - AES-128
     - Sample AES
   - The valid encryption methods for CMAF container types are:
     - CENC
     - CBCS
3. For **DRM systems**, choose the DRM system providers you’re using to protect your content during distribution. You can choose more than one. If you don't see your DRM system provider, confirm you choose the correct container type and encryption method. For supported DRM system providers, see Container and DRM system support with SPEKE (p. 35).
   - The valid DRM systems are:
     - Clear Key AES-128
     - FairPlay
     - PlayReady
     - Widevine
4. For **Resource ID**, enter an identifier for the content. The service sends this to the key server to identify the current endpoint. How unique you make this depends on how fine-grained you want access controls to be. The service does not permit you to use the same ID for two simultaneous encryption processes. The resource ID is also known as the content ID.
   - The following example shows a resource ID.
   ```
   MovieNight20171126093045
   ```
5. For **Key server URL**, enter the URL of the API Gateway proxy that you set up to talk to your key server. The API Gateway proxy must reside in the same AWS Region as MediaPackage.
   - The following example shows a URL.
   ```
   https://1wm2dx1f33.execute-api.us-west-2.amazonaws.com/SpekeSample/copyProtection
   ```
6. For **Role ARN**, enter the Amazon Resource Name (ARN) of the IAM role that provides you access to send your requests through API Gateway. Get this from your DRM solution provider.
   - The following example shows a role ARN.
   ```
   "arn:aws:iam::accountID:role/SpekeAccess
   ```
7. (Optional) For **Constant initialization vector** enter a 128-bit, 16-byte hex value represented by a 32-character string, used in conjunction with the key for encrypting content. If you don't specify a value, then MediaPackage creates the constant initialization vector (IV).
8. For **Key rotation interval (sec.)**, enter the frequency (in seconds) of key changes for live workflows, in which content is streamed real time. The service retrieves content keys before the live content begins streaming, and then retrieves them as needed over the lifetime of the workflow. By default, key rotation is 300 seconds (5 minutes), the minimum rotation interval, which is equivalent to setting it to 300. The maximum key rotation interval is 31,536,000 seconds (1 year). If you don't enter an interval, content keys aren't rotated.

The following example setting causes the service to rotate keys every thirty minutes.

```
1800
```

For information about key rotation, see [Understanding key rotation behavior](p. 36).

### Endpoint policy fields

You must assign a channel policy to enable content to flow into your channel from sources outside of your account.

1. Under **Endpoint policy**, choose an endpoint policy to enable content to flow into your channel from sources outside of your account. For more information about policies, see [Resource-based policy examples](p. 68).
   - **Don't attach a policy** - Restrict access to only those who have access to this account's credentials.
   - **Attach a custom policy** - Define your own policy and restrict access to as few or as many as you want. Enter a valid JSON object with the same structure as other IAM policies. The policy should follow the standard security advice of granting least privilege, or granting only the permissions required to perform a task.
   - **Attach a public policy** - Accept all incoming client requests to a channel's output. Enter a valid JSON object with the same structure as other IAM policies.

### Manifest fields

The manifest fields hold general information about the manifest. You must attach at least one manifest to an origin endpoint but can attach up to five manifests of one type for each origin endpoint. For example, for CMAF container types, you can attach up to five manifests for HLS and up to five manifests for LL-HLS.

1. Choose the type of manifest to use. You can choose an HLS manifest, low-latency HLS (LL-HLS) manifest, or both.
2. **For Manifest name** enter a short string that will be appended to the endpoint URL. The manifest name creates a unique path to this endpoint. If you don't enter a value, MediaPackage uses the default manifest name, `index`. MediaPackage automatically inserts the format extension, such as `.m3u8`. Supported characters are A-Z, a-z, 0-9, and - (hyphen). You can't use underscores in the name.
3. **For Child manifest name** enter a short string that will be appended to the endpoint URL. The child manifest name creates a unique path to this endpoint. Supported characters are A-Z, a-z, 0-9, and - (hyphen). You can't use underscores in the name.
4. **For Manifest window (sec.)** enter the total duration (in seconds) of the manifest's content. The maximum manifest window is 900 seconds (15 minutes).
5. **For Program date/time interval (sec.)** enter the interval (in seconds) for MediaPackage to insert the `EXT-X-PROGRAM-DATE-TIME` tags in the manifest. Program date time (PDT) is optional when using HLS manifests, but is required when using low-latency HLS manifests.

The maximum PDT interval is 1,209,600 seconds (14 days). If you don't enter an interval, `EXT-X-PROGRAM-DATE-TIME` tags aren't included in the manifest.
The EXT-X-PROGRAM-DATE-TIME tag holds the time of the segment. When PDT information is available in the source content, MediaPackage uses this same information on the output content. Otherwise, MediaPackage uses Coordinated Universal Time (UTC) for the PDT.

The PDT information helps downstream players to synchronize the stream to the wall clock, enabling functionality like viewer seek in the playback timeline and time display on the player.

6. For **Ad markers**, choose how ad markers are included in the packaged content. If you include ad markers in the content stream in your upstream encoders, then you need to inform MediaPackage what to do with the ad markers in the output. If you don’t see this field, select **Enable SCTE support** in the origin endpoint segment settings.

The valid values are:

- **Daterange** – Insert EXT-X-DATERANGE tags to signal ads and program transition events in TS and CMAF output manifests. If you choose daterange, you **must** also enter a **Program date/time interval (sec.)** value of 1 or greater.

7. For **Filter Configuration**, optionally add **Manifest filter**, **Start time**, **End time**, and **Time delay**. These filters apply to all egress requests for your endpoint.

To automatically fill these values from an existing query string, choose **Import from query string**. For example, you can import the following query string to automatically fill in **Filter key** videoCodec, **Filter value** h265, **Filter key** audioLanguage, **Filter value** fr,en-US, **Start time** 2023-10-20T12:20:50Z, **End time** 2023-10-20T13:20:50Z, and **Time delay** 10 seconds: aws.manifestfilter=videoCodec:h265;audioLanguage:fr,en-US,de&start=2023-10-20T12:20:50Z&end=2023-10-20T13:20:50Z&time_delay=10

**Manifest filter**

Optionally specify one or more manifest filters for all of your manifest egress requests.

You enter a **Filter key** and **Filter value** pair for each manifest filter. For a list of supported keys and values, see **Manifest filter query parameters** (p. 40).

For example, to restrict all manifest egress requests to 0 to 44000 Hz audio sample rate, 0 to 2147483647 video bitrate, H265 video codec, and French and English languages, enter the following key and value pairs:

- **Filter key** audioSampleRate | **Filter value**: 0-44100
- **Filter key** videoBitrate | **Filter value**: 0-2147483647
- **Filter key** videoCodec | **Filter value**: H265
- **Filter key** audioLanguage | **Filter value**: fr,en-US

**Start time and End time**

Optionally specify the start or end time for all of your manifest egress requests.

For more information about start and end times, see **Time-shifted viewing reference in AWS Elemental MediaPackage** (p. 50).

Note that if you enter a start or end time using the API, or import using **Import from query string** in the MediaPackage console, enter dates in an ISO-8601 format.

**Time delay**

Optionally specify the time delay, in seconds, for all of your manifest egress requests.

For more information about using time delays, see **Time delay** (p. 51).
**Note**

When you include a Manifest filter, you cannot use matching query parameters for the manifest's endpoint URL. If you do, you will receive a 404 HTTP error code instead. For example, if you include a Manifest filter with a `audio_sample_rate` Filter key and 44100 Filter value, and you make an HTTP request for `https://<example-url>/?aws.manifestfilter=audio_sample_rate:44100`, you will receive a 404 error.

## Viewing an origin endpoint

This guide shows how to view all origin endpoints that are configured in AWS Elemental MediaPackage. You can view the details about a specific endpoint to obtain its playback URL, the packaging settings, and the manifests within the endpoint. You can use the MediaPackage console, the AWS CLI, or the MediaPackage API to view the details of an endpoint.

### To view an origin endpoint

1. Access the channel that the endpoint is associated with, as described in [Viewing channel details (p. 24)](viewing-channel-details-24).
   
The console shows all existing origin endpoints that are configured in MediaPackage.
2. (Optional) To adjust your viewing preferences, choose **Preferences**. For example, you can adjust the page size and properties that you want to view.
3. To view more information about a specific origin endpoint, select that origin endpoint from the **Origin Endpoints** list. For downstream device requests, you must provide the endpoint URL from the **Endpoint URL** field or the CloudFront CDN URL.

## Editing an endpoint

Edit the packaging preferences on an endpoint to optimize the viewing experience. You can't change the container type after you save an endpoint or greyed-out fields. To serve content with a different packager, create a different endpoint.

If you edited the channel to enable Amazon CloudFront distribution creation from the MediaPackage console, you can also edit the endpoint to add an origin to the distribution (if you didn't already add one through alternate means). When you save the edited endpoint, MediaPackage automatically works with CloudFront to create the origin.

Any edits you make that impact the video output may not be reflected for a few minutes.

You can use the MediaPackage console, MediaPackage API, or AWS CLI to edit an origin endpoint. When you’re editing an origin endpoint, don’t put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn’t require that you supply any customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

### To edit an endpoint

1. Access the channel that the endpoint is associated with, as described in [Viewing channel details (p. 24)](viewing-channel-details-24).
   
The console shows all existing origin endpoints that are configured in MediaPackage.
2. Under **Origin endpoints**, choose the endpoint that you want to edit and then choose **Edit endpoint**.
3. Edit the endpoint options that you want to change.
4. Choose **Edit**.
Deleting an endpoint

Endpoints can serve content until they’re deleted. This guide shows how to delete the endpoint if it should no longer respond to playback requests. You must delete all endpoints from a channel before you can delete the channel.

**Warning**
If you delete an endpoint, the playback URL stops working.

You can use the MediaPackage console, the AWS CLI, or the MediaPackage API to delete an endpoint.

**To delete an endpoint**

1. Access the channel that the endpoint is associated with, as described in Viewing channel details (p. 24).
   - The console shows all existing origin endpoints that are configured in MediaPackage.
2. Under Origin endpoints, choose the endpoint that you want to delete.
3. Choose Delete.
4. In the Delete endpoints confirmation dialog box, choose Delete.

Previewing a manifest

Preview an endpoint’s manifest to ensure that MediaPackage is receiving the content stream and can package it. The preview is helpful for avoiding playback failures after the endpoint is published and for troubleshooting later if there are any playback issues.

You can use the MediaPackage console to preview playback from the endpoint.

**To preview an endpoint’s playback**

1. Access the channel that the endpoint is associated with, as described in Viewing channel details (p. 24).
2. Under Origin endpoints, select the endpoint that you want to preview.
3. To preview playback, do one of the following:
   - Choose Preview to play content with the embedded player.
   - Choose QR code to view and scan the QR code for playback on a compatible device.
Delivering VOD content from AWS Elemental MediaPackage

At this time, MediaPackage v2 doesn't support video on demand (VOD) or live-to-VOD workflows. If you're looking to support VOD workflows, see the AWS Elemental MediaPackage v1 User Guide.
Content encryption and DRM in AWS Elemental MediaPackage

Protect your content from unauthorized use through content encryption and digital rights management (DRM). AWS Elemental MediaPackage uses the AWS Secure Packager and Encoder Key Exchange (SPEKE) API to facilitate content encryption and decryption by a DRM provider. Using SPEKE, the DRM provider supplies encryption keys to MediaPackage through the SPEKE API. The DRM provider also supplies licenses to supported media players for decryption. For more information about how SPEKE is used with services and features running in the cloud, see AWS cloud-based architecture in the Secure Packager and Encoder Key Exchange API Specification guide.

Limitations and requirements

When implementing content encryption for MediaPackage, refer to the following limitations and requirements:

- Use the AWS Secure Packager and Encoder Key Exchange (SPEKE) API to facilitate integration with a digital rights management (DRM) system provider. For information about SPEKE, see What is Secure Packager and Encoder Key Exchange?
- Your DRM system provider must support SPEKE. For a list of DRM providers that support SPEKE, see the Get on board with a DRM platform provider topic in the AWS Elemental MediaPackage User Guide. Your DRM provider can help you set up DRM encryption use in MediaPackage.
- Use MediaPackage to encrypt live content.

The following sections provide guidance on how to choose and implement content encryption using SPEKE for MediaPackage.

Topics

- Content encryption and DRM in AWS Elemental MediaPackage (p. 34)
- Manifest filtering (p. 38)
- Metadata passthrough (p. 45)
- Rendition groups reference in AWS Elemental MediaPackage (p. 45)
- SCTE-35 message options in AWS Elemental MediaPackage (p. 46)
- Time-shifted viewing reference in AWS Elemental MediaPackage (p. 50)
- Working with trick-play in AWS Elemental MediaPackage (p. 52)
- Working with data plane APIs in AWS Elemental MediaPackage (p. 53)
Container and DRM system support with SPEKE

MediaPackage supports SPEKE Version 2.0 which uses multiple, distinct encryption keys for audio and video tracks and uses Content Protection Information Exchange (CPIX) Version 2.3. For more information about SPEKE Version 2.0 encryption configurations, see SPEKE Version 2.0 presets (p. 36).

Supported containers and DRM systems

The following table lists the different containers and digital rights management (DRM) systems that SPEKE Version 2.0 supports.

<table>
<thead>
<tr>
<th>SPEKE Version 2.0 – Support matrix for container and DRM system</th>
<th>Apple FairPlay</th>
<th>ClearKey AES-128</th>
<th>Google Widevine</th>
<th>Microsoft PlayReady</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS container</td>
<td>✓</td>
<td>✓</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>Supports SAMPLE-AES</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supports AES-128</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMAF container</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Supports cbcs encryption</td>
<td></td>
<td>Not supported</td>
<td></td>
<td>Supports cbcs and cenc encryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Supports cbcs and cenc encryption</td>
</tr>
</tbody>
</table>

Supported DRM system IDs

The following table lists the different DRM system IDs that MediaPackage supports.

<table>
<thead>
<tr>
<th>System IDs – Support matrix for DRM system</th>
<th>Apple FairPlay</th>
<th>ClearKey AES-128</th>
<th>Google Widevine</th>
<th>Microsoft PlayReady</th>
</tr>
</thead>
<tbody>
<tr>
<td>94ce86fb-07ff-4f43-adb8-93d2fa968ca2</td>
<td>3ea8778f-7742-4bf9edef8ba9-79d6-4ace9a04f079-9840-42b6-a3c8-27dcd51d21edab92-e65be0885f95</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Deploying SPEKE

Your digital rights management (DRM) system provider can help you get set up to use DRM encryption in MediaPackage. Generally, the provider gives you a SPEKE gateway to deploy in your AWS account in the same AWS Region where MediaPackage is running. For information about configuring encryption settings for your endpoint, see encryption fields.

If you must build your own API Gateway to connect MediaPackage to your key service, you can use the SPEKE Reference Server available on GitHub as a starting point.
Understanding key rotation behavior

When you enable key rotation on live content from TS and CMAF origin endpoints, AWS Elemental MediaPackage retrieves content keys before the live content begins. As the content progresses, MediaPackage retrieves new keys at the interval that you set on the origin endpoint, as described in Encryption fields (p. 27).

If MediaPackage is unable to retrieve the content key, it takes the following actions:

- If MediaPackage successfully retrieved a content key for this endpoint before, it uses the last key that it fetched. This ensures that endpoints that worked previously continue to work.
- If MediaPackage has not successfully retrieved a content key for this endpoint before, MediaPackage responds to the playback request with error 404.

SPEKE Version 2.0 presets

SPEKE Version 2.0 supports the use of multiple, distinct encryption keys for audio and video tracks. MediaPackage uses presets to configure the encryption. The MediaPackage API defines these presets, and they appear in the MediaPackage console in the Video encryption preset and Audio encryption preset menus of the Package Encryption endpoints configuration section. The presets map encryption keys to specific audio or video tracks, based on the number of channels for audio tracks, and based on the video resolution for video tracks. MediaPackage uses specific combinations of audio and video encryption presets to support three different encryption scenarios:

- Scenario 1: Unencrypted tracks and encrypted tracks (p. 36)
- Scenario 2: Single encryption key for all audio and video tracks (p. 36)
- Scenario 3: Multiple encryption keys for audio and video tracks (p. 36)

Scenario 1: Unencrypted tracks and encrypted tracks

You can choose not to encrypt the audio or the video tracks by selecting the UNENCRYPTED preset in the Video encryption preset or the Audio encryption preset menus. You can't select UNENCRYPTED for both audio and video presets, because doing so would mean that you don't intend to encrypt any of the tracks at all. Also, you can't combine UNENCRYPTED and SHARED presets for audio and video, because SHARED is a special preset. For more information, see Scenario 2: Single encryption key for all audio and video tracks (p. 36).

The following list describes valid combinations of UNENCRYPTED presets:

- UNENCRYPTED for audio tracks, and any video preset with a name that starts with PRESET-VIDEO-
- UNENCRYPTED for video tracks, and any audio preset with a name that starts with PRESET-AUDIO-

Scenario 2: Single encryption key for all audio and video tracks

The SPEKE Version 2.0 SHARED preset uses a single encryption key for all audio and video tracks, as in SPEKE Version 1.0. When you select the SHARED preset, select it for both audio and video encryption.

Scenario 3: Multiple encryption keys for audio and video tracks

When you use a preset with a name that starts with PRESET-VIDEO- or PRESET-AUDIO-, MediaPackage encrypts the audio tracks and video tracks with the number of encryption keys that the specific preset defines. The following tables show how many keys MediaPackage requests from
the key server and how those keys map to tracks. If no track matches the criteria for a particular key, MediaPackage does not use that key to encrypt any track.

MediaPackage encrypts I-frame only trickplay tracks with the key corresponding to their resolution.

In the following table, the Key name value is the value of the ContentKeyUsageRule@IntendedTrackType attribute that MediaPackage uses in the CPIX document. This is sent to the SPEKE server for a specific content key.

**Video encryption presets**

<table>
<thead>
<tr>
<th>Preset name</th>
<th>Number of keys</th>
<th>Key name</th>
<th>Minimum resolution</th>
<th>Maximum resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRESET-VIDEO-1</td>
<td>1</td>
<td>VIDEO</td>
<td>No minimum or maximum resolution. MediaPackage encrypts all tracks with the same key.</td>
<td></td>
</tr>
<tr>
<td>PRESET-VIDEO-2</td>
<td>2</td>
<td>SD</td>
<td>No minimum</td>
<td>&lt;= 1024x576</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HD</td>
<td>&gt; 1024x576</td>
<td>No maximum</td>
</tr>
<tr>
<td>PRESET-VIDEO-3</td>
<td>3</td>
<td>SD</td>
<td>No minimum</td>
<td>&lt;= 1024x576</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HD</td>
<td>&gt; 1024x576</td>
<td>&lt;= 1920x1080</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UHD</td>
<td>&gt; 1920x1080</td>
<td>No maximum</td>
</tr>
<tr>
<td>PRESET-VIDEO-4</td>
<td>4</td>
<td>HD</td>
<td>&gt; 1024x576</td>
<td>&lt;= 1920x1080</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UHD1</td>
<td>&gt; 1920x1080</td>
<td>&lt;= 4096x2160</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UHD2</td>
<td>&gt; 4096x2160</td>
<td>No maximum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SD</td>
<td>No minimum</td>
<td>&lt;= 1024x576</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HD1</td>
<td>&gt; 1024x576</td>
<td>&lt;= 1280x720</td>
</tr>
<tr>
<td>PRESET-VIDEO-5</td>
<td>5</td>
<td>HD2</td>
<td>&gt; 1280x720</td>
<td>&lt;= 1920x1080</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UHD1</td>
<td>&gt; 1920x1080</td>
<td>&lt;= 4096x2160</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UHD2</td>
<td>&gt; 4096x2160</td>
<td>No maximum</td>
</tr>
<tr>
<td>PRESET-VIDEO-6</td>
<td>4</td>
<td>SD</td>
<td>No minimum</td>
<td>&lt;= 1024x576</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HD1</td>
<td>&gt; 1024x576</td>
<td>&lt;= 1280x720</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HD2</td>
<td>&gt; 1280x720</td>
<td>&lt;= 1920x1080</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UHD</td>
<td>&gt; 1920x1080</td>
<td>No maximum</td>
</tr>
<tr>
<td>PRESET-VIDEO-7</td>
<td>3</td>
<td>SD+HD1</td>
<td>No minimum</td>
<td>&lt;= 1280x720</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HD2</td>
<td>&gt; 1280x720</td>
<td>&lt;= 1920x1080</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UHD</td>
<td>&gt; 1920x1080</td>
<td>No maximum</td>
</tr>
<tr>
<td>PRESET-VIDEO-8</td>
<td>4</td>
<td>SD+HD1</td>
<td>No minimum</td>
<td>&lt;= 1280x720</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HD2</td>
<td>&gt; 1280x720</td>
<td>&lt;= 1920x1080</td>
</tr>
</tbody>
</table>
### Preset Table

<table>
<thead>
<tr>
<th>Preset name</th>
<th>Number of keys</th>
<th>Key name</th>
<th>Minimum resolution</th>
<th>Maximum resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>UHD1</td>
<td></td>
<td></td>
<td>&gt; 1920x1080</td>
<td>&lt;= 4096x2160</td>
</tr>
<tr>
<td>UHD2</td>
<td></td>
<td></td>
<td>&gt; 4096x2160</td>
<td>No maximum</td>
</tr>
<tr>
<td>SHARED</td>
<td>1</td>
<td>ALL</td>
<td>No minimum or maximum resolution. MediaPackage encrypts all video and audio tracks with the same key.</td>
<td></td>
</tr>
<tr>
<td>UNENCRYPTED</td>
<td>0</td>
<td>N/A</td>
<td>MediaPackage does not encrypt any video track.</td>
<td></td>
</tr>
</tbody>
</table>

In the following table, the **Key name** value is the value of the `ContentKeyUsageRule@IntendedTrackType` attribute that MediaPackage uses in the CPIX document. This is sent to the SPEKE server for a specific content key.

#### Audio encryption presets

<table>
<thead>
<tr>
<th>Preset name</th>
<th>Number of keys</th>
<th>Key name</th>
<th>Minimum number of channels</th>
<th>Maximum number of channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRESET-AUDIO-1</td>
<td>1</td>
<td>AUDIO</td>
<td>No minimum or maximum number of channels. MediaPackage encrypts all audio and video tracks with the same key.</td>
<td></td>
</tr>
<tr>
<td>PRESET-AUDIO-2</td>
<td>2</td>
<td>STEREO_AUDIO</td>
<td>No minimum</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MULTICHANNEL_AUDIO</td>
<td>&gt; 2</td>
<td>No maximum</td>
</tr>
<tr>
<td>PRESET-AUDIO-3</td>
<td>3</td>
<td>STEREO_AUDIO</td>
<td>No minimum</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MULTICHANNEL_AUDIO_3_6</td>
<td>&gt; 2</td>
<td>&lt;= 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MULTICHANNEL_AUDIO_7</td>
<td>&gt; 6</td>
<td>No maximum</td>
</tr>
<tr>
<td>SHARED</td>
<td>1</td>
<td>ALL</td>
<td>No minimum or maximum number of channels. MediaPackage encrypts all audio and video tracks with the same key.</td>
<td></td>
</tr>
<tr>
<td>UNENCRYPTED</td>
<td>0</td>
<td>N/A</td>
<td>MediaPackage does not encrypt any audio track.</td>
<td></td>
</tr>
</tbody>
</table>

Now you know how MediaPackage supports SPEKE Version 2.0 presets for unencrypted tracks and encrypted tracks. With these presets, you can use a single encryption key for all audio and video tracks, and multiple encryption keys for audio and video tracks.

### Manifest filtering

With manifest filtering, AWS Elemental MediaPackage dynamically produces client manifests based on parameters that you specify in a query appended to your playback request. This enables you to do things such as restrict viewer access to premium 4K HEVC content, or target specific device types and audio sample rate ranges, all from a single endpoint. Previously, you would have to configure multiple.
endpoints to accomplish this behavior. MediaPackage now provides a cost-effective way to dynamically produce different client manifests on the same endpoint.

## Working with manifest filters

When you use a manifest filter, the resulting manifest includes only the audio and video streams that match the characteristics that you specify in your query. If no manifest filter is used, then all of the ingested streams are present in the endpoint output stream. The exception to this is if you have set stream filters for the endpoint, such as minimum video bitrate. In that case, the manifest filter is applied after the stream filter, which could skew your output, and is not recommended.

Manifest filtering can be used on all origin endpoint types supported by MediaPackage:

- TS
- CMAF

To use manifest filtering, append `aws.manifestfilter` query parameters to your playback request to MediaPackage. MediaPackage evaluates the query, and serves a client manifest based on those query parameters. Manifest queries are not case-sensitive and can be up to 1024 characters long. If the query is malformed, or if it there aren’t streams that match the query parameters, MediaPackage returns an incomplete or empty manifest. For query syntax, see the following section.

**Note**

If you are using TS or CMAF origin endpoints, special conditions apply. For information about these conditions, see [Special conditions for TS and CMAF manifests](p. 43).

### Query syntax

The base query parameter is `aws.manifestfilter`, which is followed by optional parameter name and value pairs. To construct the query, append `?aws.manifestfilter=` to the end of the MediaPackage endpoint URL, followed by parameter names and values. For a list of all of the available parameters, see [Manifest filter query parameters](p. 40).

An Apple HLS filter query might look like this:

```text
https://example-mediapackage-endpoint.mediapackage.us-west-2.amazonaws.com/out/v1/examplemediapackage/index.m3u8?
aws.manifestfilter=audio_sample_rate:0-44100;video_bitrate:0-2147483647;video_codec:h265;audio_language:fr,en-US,de
```

The query syntax is listed in the following table.

<table>
<thead>
<tr>
<th>Query string component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>?</code></td>
<td>A restricted character that marks the beginning of a query.</td>
</tr>
<tr>
<td><code>aws.manifestfilter=</code></td>
<td>The base query, which is followed by parameters constructed of name and value pairs. For a list of all of the available parameters, see [Manifest filter query parameters](p. 40).</td>
</tr>
<tr>
<td><code>:</code></td>
<td>Used to associate the parameter name with a value. For example, <code>parameter_name:value</code>.</td>
</tr>
<tr>
<td><code>;</code></td>
<td>Separates parameters in a query that contains multiple parameters. For example, <code>parameter1_name:value;parameter2_name:minValue-maxValue</code>.</td>
</tr>
</tbody>
</table>
Query string component | Description
---|---
, | Separates a list of values. For example, `parameter_name: value1,value2,value3`. Comma-separated values in a list imply an OR relationship.
- | Used to define a parameter’s minimum - maximum value range. For example, `audio_sample_rate:0-44100`. When a numerical value is used in a range, it is included in the range definition. This means that streams must be greater than or equal to the minimum value, and less than or equal to the maximum value. With ranges, the minimum and maximum values are mandatory. The supported range values are 0 - 2147483647.

**Note**
If you use Amazon CloudFront as your CDN, you might need to set additional configurations. For more information, see [Configure cache behavior for all endpoints](#).

### Manifest filter query parameters

MediaPackage supports the following query parameters.

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio</td>
<td>audio_bitrate</td>
<td>• The audio bitrate in bits per second.</td>
<td><code>stream.mpd?aws.manifestfilter=audio_bitrate:0-2147483647</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>Accepted values</strong>: Two integers aggregated with a dash that define an inclusive range. The supported range values are 0 - 2147483647.</td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>audio_channels</td>
<td>• The number of audio channels.</td>
<td><code>stream.mpd?aws.manifestfilter=audio_channels:1-8</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>Accepted values</strong>: Two integers aggregated with a dash that define an inclusive range. The supported range values are 1 - 32767.</td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>audio_codec</td>
<td>• The audio codec type.</td>
<td><code>stream.mpd?aws.manifestfilter=audio_codec:AACL,AC-3</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>Accepted values</strong>: AACL, AACH, AC-3, EC-3. You must include the - for AC-3 and EC-3. The values are not case-sensitive.</td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>audio_language</td>
<td>• Audio languages or functional codes derived from encoder passthrough.</td>
<td><code>stream.mpd?aws.manifestfilter=audio_language:fr,en-US,de</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>Accepted values</strong>: Arbitrary strings, such as two or four character ISO-639-1 language codes. You must use the same language strings that are set for your encoder. The values are not case-sensitive.</td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>audio_sample_rate</td>
<td>• The audio sample rate in Hz.</td>
<td><code>stream.mpd?aws.manifestfilter=audio_sample_rate:0-44100</code></td>
</tr>
</tbody>
</table>
### Manifest filter query parameters

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Accepted values:</strong> Two integers aggregated with a dash that define an inclusive range. The supported range values are ( 0 - 2147483647 ).</td>
<td></td>
</tr>
<tr>
<td>Subtitle</td>
<td>subtitle_language</td>
<td>• The subtitle language or functional codes derived from encoder passthrough.</td>
<td>stream.mpd? aws.manifestfilter=subtitle_language:en-US, hi</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>Accepted values:</strong> Arbitrary strings, such as two or four character ISO-639-1 language codes. You must use the same language strings that are set for your encoder.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The values are not case-sensitive.</td>
<td></td>
</tr>
<tr>
<td>Video</td>
<td>trickplay_height</td>
<td>• The height of the trick-play image in pixels. This applies to both I-frame only and image-based trick-play.</td>
<td>stream.mpd? aws.manifestfilter=trickplay_height:200-1200</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you're using this parameter with I-frame only trick-play, trickplay_height and video_height should have similar values. If the values are not the same, I-frame only tracks might be removed from a manifest.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>Accepted values:</strong> Two integers aggregated with a dash that define an inclusive range. The supported range values are ( 1 - 2147483647 ).</td>
<td></td>
</tr>
<tr>
<td>Video</td>
<td>trickplay_type</td>
<td>• The trickplay track type.</td>
<td>stream.mpd? aws.manifestfilter=trickplay_type:default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>Accepted values:</strong> iframe, image, none.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The values are not case-sensitive.</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Name</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Video</td>
<td>video_bitrate</td>
<td>• The video bitrate in bits per second.</td>
<td>stream.mpd? aws.manifestfilter=video_bitrate:0-2147483647</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you're using this parameter, we recommend that you use only the video_bitrate filter parameter to set the video bitrate. Don't also set the minimum and maximum video bitrate via the MediaPackage console or AWS CLI. The video_bitrate filter applies to the video bitrate settings created at the endpoint. If you use the parameter and set the bitrate in the console or AWS CLI, your output might be skewed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Accepted values: Two integers aggregated with a dash that define an inclusive range. The supported range values are 0 - 2147483647.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>Note</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>You can't use this parameter with trick-play streams.</td>
<td></td>
</tr>
<tr>
<td>Video</td>
<td>video_codec</td>
<td>• The video codec type.</td>
<td>stream.mpd? aws.manifestfilter=video_codec:h264</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Accepted values: H264, H265.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The values are not case-sensitive.</td>
<td></td>
</tr>
<tr>
<td>Video</td>
<td>video_dynamic_range</td>
<td>• The video dynamic range.</td>
<td>stream.mpd? aws.manifestfilter=video_dynamic_range:hdr10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Accepted values: hdr10, hlg, sdr.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The values are not case-sensitive.</td>
<td></td>
</tr>
<tr>
<td>Video</td>
<td>video_framerate</td>
<td>• The video frame rate range in the NTSC format.</td>
<td>stream.mpd? aws.manifestfilter=video_framerate:23.976-30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Accepted values: Two floating-point numbers aggregated with a dash that define an inclusive range. Each number can have up to three optional fractional values. For example, 29.97 or 29.764. The supported range values are 1 - 999.999.</td>
<td></td>
</tr>
</tbody>
</table>
### Manifest filtering examples

#### Example 1: Target a player that supports AVC and a 44.1k audio sample rate

The viewer is playing content on a device that can only support AVC and a 44.1k audio sample rate. You set the `video_codec` and `audio_sample_rate` to filter out streams that don't fit these requirements.

?aws.manifestfilter=audio_sample_rate:0-44100;video_codec:h264

#### Example 2: Restrict 4k HEVC content

Your 4K HEVC stream is 15 Mbps, and all your other streams are less than 9 Mbps. To exclude the 4K stream from the stream set, you set a threshold of 9,000,000 bits per second to filter out the higher bitrate.

?aws.manifestfilter=video_bitrate:0-9000000

#### Example 3: Include video between 23.976 and 30 frames per second

To only include video within a certain frame rate range, use `video_framerate`. This parameter accepts floating-point numbers with up to three optional decimal values.

?aws.manifestfilter=video_framerate:23.976-30

### Special conditions for TS and CMAF manifests

If you are using TS or CMAF manifests, these special conditions apply.
• For TS manifests, we strongly recommend that you use audio rendition groups to avoid removing the video streams that are multiplexed with the audio streams that are filtered out. For more information about rendition groups, see Rendition groups reference in AWS Elemental MediaPackage (p. 45).

• In TS and CMAF manifests, the audio sample rate is not signaled, so it's not easy to visually check the original or filtered manifests for this setting. To verify the audio sample rate, check the audio sample rate at the encoder level and output level.

• In TS and CMAF manifests, the BANDWIDTH attribute for a variant associates the bandwidth of the audio track with the video track, whether it is multiplexed with the video track, or if it is an audio rendition track referenced by the video track. Therefore, you can't visually inspect the original and filtered manifests to confirm the video_bitrate filter has worked. To verify the filter, check the video bitrate at the encoder level and output level.

• For TS and CMAF manifests, request parameters appended to bitrate playlists or segments result in an HTTP 400 error.

## Error conditions

Common error conditions are listed in the following table.

<table>
<thead>
<tr>
<th>Error condition</th>
<th>Example</th>
<th>HTTP status code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A list parameter is not found and is not part of a constrained list</td>
<td>aws.manifestfilter=audio_language:dahlia</td>
<td>200</td>
</tr>
<tr>
<td>Only subtitle streams are present in the stream</td>
<td>aws.manifestfilter=audio_sample_rate:0-1;video_bitrate=0-1</td>
<td>200</td>
</tr>
<tr>
<td>Duplicate filter parameter</td>
<td>aws.manifestfilter=audio_sample_rate:0-48000</td>
<td>400</td>
</tr>
<tr>
<td>Invalid parameter</td>
<td>aws.manifestfilter=audio_language:rhododendron</td>
<td>400</td>
</tr>
<tr>
<td>Invalid range parameter</td>
<td>aws.manifestfilter=audio_sample_rate:300-0</td>
<td>400</td>
</tr>
<tr>
<td>Invalid range value (more than INT_MAX)</td>
<td>aws.manifestfilter=audio_sample_rate:0-2147483648</td>
<td>400</td>
</tr>
<tr>
<td>Malformed query string</td>
<td>aws.manifestfilter=audio_language:is:0-44100</td>
<td>400</td>
</tr>
<tr>
<td>Parameter string is greater than 1024 characters</td>
<td>aws.manifestfilter=audio_language:abcdef....</td>
<td>400</td>
</tr>
<tr>
<td>Query parameters on an TS or CMAF bitrate manifest</td>
<td>index_1.m3u8?</td>
<td>400</td>
</tr>
<tr>
<td>Query parameters on a segment request</td>
<td>..._1.ts</td>
<td>mp4</td>
</tr>
<tr>
<td>Repeated query parameter</td>
<td>aws.manifestfilter=audio_sample_rate:0-48000</td>
<td>400</td>
</tr>
<tr>
<td>Application of the filter results in an empty manifest (content has no streams that meet the)</td>
<td>aws.manifestfilter=audio_sample_rate:0-1;video_bitrate=0-1</td>
<td>400</td>
</tr>
</tbody>
</table>
Metadata passthrough

AWS Elemental MediaPackage automatically passes through ID3 metadata from a channel's input to the channel's output stream. You don't need to adjust your endpoint's configuration to enable metadata passthrough.

ID3 metadata considerations

Timed ID3 metadata is a general-purpose mechanism that adds synchronized metadata to streams. The metadata is used for a variety of purposes, ranging from interactive applications to audience measurement.

Supported MediaPackage endpoint types

MediaPackage supports ID3 metadata passthrough for the following endpoint types:

- Live TS and CMAF origin endpoints

Metadata carriage

Here is how ID3 is carried as metadata in the following specifications:

- TS - Metadata is carried in the elementary stream. For more information, see section 2.0 of the Apple Timed Metadata for HTTP Live Streaming reference.
- CMAF - Metadata is carried in the Event Message box version 1. For more information, see Carriage of ID3 Timed Metadata in CMAF. Event Message boxes include a scheme_id_uri field set to https://aomedia.org/emsg/ID3 and a value field set to 0.

Metadata signaling

HLS manifests don't have specific metadata signaling.

MediaLive configuration

You can produce ID3 metadata in AWS Elemental MediaLive MediaPackage output groups either by passing through ID3 metadata, or inserting ID3 metadata using the schedule.

Rendition groups reference in AWS Elemental MediaPackage

Rendition groups are used in TS and CMAF outputs. A rendition group collects all subtitle or audio tracks and makes them available for all video renditions in the stream. When you enable rendition groups, MediaPackage pulls together all audio variants (such as different languages or codecs) and groups them for use with any video rendition. MediaPackage automatically puts subtitles into a rendition group.

Audio and subtitles tracks are required to be in their own rendition groups for CMAF outputs.
The following sections further describe when you can use rendition groups.

**When to use rendition groups**

Rendition groups are used only in TS and CMAF outputs. Rendition groups are most beneficial when you have multiple languages or multiple audio codecs in your streams. Rendition groups should be used in the following use cases:

- With CMAF outputs, if there are any audio or subtitle tracks
  
  CMAF requires all audio tracks in one rendition group, and all subtitles in another. Audio or subtitles can't be muxed with video tracks.
- One or more video tracks with multiple audio languages or codecs
  
  When rendition groups are enabled, MediaPackage pulls all audio renditions together for shared use between the video tracks. In this way, you don't have to duplicate all the audio options across all the video tracks.
- Multiple audio-only tracks and multiple subtitle tracks
  
  When both the audio tracks and subtitle tracks are in rendition groups, all the audio options can be combined with any subtitle track.
- One audio-only track and multiple subtitle tracks
  
  MediaPackage automatically pulls subtitle tracks into a rendition group so that the audio track can be used with any subtitle. Because there is only one audio and the subtitles are already grouped, you don't need to tell MediaPackage to use rendition groups in this case.

**When not to use rendition groups**

Rendition groups can't or shouldn't be used in the following use cases:

- Multiple video tracks in the stream, but only one language or codec is used for the audio. If the same audio is used with multiple video tracks, and rendition groups are also used, then your rendition group will have duplicates of the same audio track (one for each video).
  
  Keep the audio and video muxed in the stream, and do not use a rendition group.

**SCTE-35 message options in AWS Elemental MediaPackage**

This section describes the options that AWS Elemental MediaPackage offers for configuring how SCTE-35 messages are handled in live TS and CMAF outputs.

SCTE-35 messages accompany video in your source content. These messages signal where MediaPackage should insert ad markers when it packages the content for output. By default, MediaPackage inserts markers for the following message types in the source content:

- Splice insert
- Break
- Provider advertisement
- Distributor advertisement
• Provider placement opportunity
• Distributor placement opportunity
• Provider overlay placement opportunity
• Distributor overlay placement opportunity
• Program

When these commands are present, MediaPackage inserts corresponding ad markers in the output manifests:

• For daterange on TS and CMAF origin endpoints, MediaPackage inserts EXT-X-DATERANGE tags.

The following sections describe how you can modify MediaPackage SCTE-35 message handling behavior.

**SCTE-35 settings in MediaPackage**

You can modify how MediaPackage interacts with SCTE-35 messages from your source content. Configure the following settings on your origin endpoints. For more information, see the following:

• For the MediaPackage console, see the section called “Creating an origin endpoint” (p. 26).
• For the MediaPackage API, see CreateOriginEndpoint in the AWS Elemental MediaPackage Live API Reference.

**Important**

To modify how MediaPackage handles SCTE-35 messages, you should be familiar with the SCTE-35 standard. You can view the most recent standards here: SCTE Standards Catalog. You should also be familiar with how SCTE-35 is implemented in your source content.

**Ad markers**

This setting is available on TS and CMAF origin endpoints.

**Ad markers** allows you to specify what MediaPackage does when it detects SCTE-35 messages. These are the options:

• **Daterange** – Insert EXT-X-DATERANGE tags to signal ads and program transition events in TS and CMAF output manifests. If you choose daterange, you must also enter a Program date/time interval (sec.) value of 1 or greater.

**SCTE filtering**

This setting is available on TS and CMAF origin endpoints.

**SCTE filtering** identifies which SCTE-35 message types MediaPackage treats as ads in the output manifest.

If you don't change this setting, MediaPackage treats these message types as ads:

• Splice insert
• Break
• Provider advertisement
• Distributor advertisement
• Provider placement opportunity
• Distributor placement opportunity
• Provider overlay placement opportunity
How it works

The Ad markers and SCTE filtering settings work together to determine what MediaPackage does with SCTE-35 messages from the source content.

When there are SCTE-35 messages in the source content, MediaPackage takes the following action based on the value that you selected in Ad markers:

- For Daterange, MediaPackage inserts EXT-X-DATERANGE tags to signal ads and program transition events in TS and CMAF output manifests.

EXT-X-DATERANGE ad markers

Daterange ad markers are used to signal ads and program transitions in live TS and CMAF manifests. When you enable daterange ad markers on your origin endpoint, MediaPackage inserts EXT-X-DATERANGE tags into the manifest where there are SCTE-35 time_signal or splice_insert tags present. EXT-X-DATERANGE is used in concert with EXT-X-PROGRAM-DATE-TIME tags.


Enabling daterange via the console

To enable daterange ad markers when creating or editing an origin endpoint, in the MediaPackage console, under the manifest settings, SCTE configuration, Ad markers, choose Daterange.

If you choose daterange, you must also enter a Program date/time interval (sec.) value of 1 or greater. The program date/time interval is set in the same manifest fields as the ad marker settings.

Enabling daterange via the AWS CLI

To enable daterange ad markers for your origin endpoint, run the following command in the AWS CLI replacing region with your own information:

```bash
aws --endpoint=https://mediapackagev2.region.amazonaws.com mediapackage --region region create-origin-endpoint --channel-id test_channel --id hlsmuxed --hls-package "{"ProgramDateTimeIntervalSeconds":60,"AdMarkers":"DATERANGE"}"
```

Important

You must set a ProgramDateTimeIntervalSeconds value that's greater than 0 (zero).

Enabling daterange via the MediaPackage API or AWS SDK

To learn how to enable daterange ad markers for TS and CMAF origin endpoints via the MediaPackage live API or AWS SDK, see the following:

- MediaPackage Live API reference
- AWS SDK
Example TS manifest showing SCTE-35 EXT-X-DATERANGE signaling

This example TS manifest generated by MediaPackage uses EXT-X-DATERANGE and EXT-X-PROGRAM-DATE-TIME tags to signal events in the live stream.

**Note**
The DURATION, PLANNED-DURATION, and END-DATE attributes of the EXT-X-DATERANGE tag are optional. If these attributes aren't present in the SCTE-35 input, or aren't set when you create your origin endpoint via the MediaPackage API, then they are omitted from the generated manifests.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:8
#EXT-X-MEDIA-SEQUENCE:11
#EXT-X-DATERANGE:ID="2415919105",START-DATE="2020-05-03T00:01:00.018Z",PLANNED-DURATION=29.988,SCTE35-OUT=0xFC303000000002CDE4000FF00506FE00526C14001A0218435545499000000817FC0000292EA80A4ABC0001300000D6F17117
#EXT-X-DATERANGE:ID="2147483649",START-DATE="2020-05-03T00:00:30.030Z",PLANNED-DURATION=90.006,SCTE35-CMD=0xFC303000000002CDE4000FF00506FE00293D6C001A021843554549800000017FFF000789ABC0A4ABCD0001100000680
#EXT-X-PROGRAM-DATE-TIME:2020-05-03T00:01:08.040Z
#EXTINF:7.560, index_1_11.ts?m=1588607409
#EXTINF:7.560, index_1_12.ts?m=1588607409
#EXTINF:6.846, index_1_13.ts?m=1588607409
#EXT-X-DATERANGE:ID="2415919105",START-DATE="2020-05-03T00:01:00.018Z",END-DATE="2020-05-03T00:01:30.006Z",DURATION=29.988
#EXTINF:0.714, index_1_14.ts?m=1588607409
#EXTINF:7.560, index_1_15.ts?m=1588607409
#EXTINF:7.560, index_1_16.ts?m=1588607409
#EXTINF:6.636, index_1_17.ts?m=1588607409
#EXT-X-DATERANGE:ID="2147483649",START-DATE="2020-05-03T00:00:30.030Z",END-DATE="2020-05-03T00:01:30.006Z",DURATION=90.006,SCTE35-CMD=0xFC304A00000002CDE4000FF00506FE00A4D82800340218435545498000000017FC000000000000A4ABCD00011100000618
#EXT-X-DATERANGE:ID="2147483650",START-DATE="2020-05-03T00:02:00.036Z",PLANNED-DURATION=90.006,SCTE35-CMD=0xFC304A00000002CDE4000FF00506FE00A4D82800340218435545498000000017FC000000000000A4ABCD00011100000618
#EXTINF:0.924, index_1_19.ts?m=1588607409
#EXTINF:7.560, index_1_20.ts?m=1588607409
#EXT-X-PROGRAM-DATE-TIME:2020-05-03T00:02:08.520Z
#EXTINF:7.560, index_1_21.ts?m=1588607409
#EXT-X-ENDLIST
```
Time-shifted viewing reference in AWS Elemental MediaPackage

Time-shifted viewing is available with live workflows in AWS Elemental MediaPackage.

*Time-shifted viewing* means that viewers can start watching a live stream at a time earlier than "now," permitting them to join from the beginning a show that's already in progress or to watch a show that's already completed. MediaPackage supports time-shifted viewing for content that's up to 336 hours (14 days) old. You can enable time-shifted viewing for some or all of this content by defining the **startover window** on the endpoint. Content that falls within that window is available for playback when playback requests include valid start and end parameters.

In the following steps, "now" is determined either by the program date time (PDT) present in the source content from the encoder or, if this PDT information is not included, by the MediaPackage ingest time of the most recent segment.

**To enable time-shifted viewing**

1. Enable time-shifted viewing by typing a value for **Startover window** on the MediaPackage endpoint object. You can do this through either the MediaPackage console or the MediaPackage API.

   When requests with start and end parameters that are within the startover window are sent to this endpoint, MediaPackage generates a manifest for the requested timeframe. If the start parameter is outside of the startover window, or if the end parameter is before the startover window, the playback request fails. If no start and end parameters are used, the service generates a standard manifest.

   **Note**
   
   You might notice that the manifest lags behind real time when you initially create a startover window on an endpoint. This is because MediaPackage starts filling the manifest from the start of the window, and works up to "now." So, if you have a 24-hour startover window, MediaPackage fills the manifest starting 24 hours ago and working up to "now."

2. Ensure that content requests contain start and end parameters as needed. MediaPackage accepts requests for up to 24 hours of content.

   For packager-specific rules about how you can notate the parameters, see [Rules for start and end parameters](#) (p. 51).

The start and end parameters determine the time boundaries of the manifest. These are the expected behaviors based on request start and end parameters:

- If both start and end parameters are specified, the resulting manifest has a fixed start and end time that correspond to the specified start and end parameters.

  If the end time is in the future, the tags in the manifest are consistent with an event manifest. The manifest will continue to grow until it reaches the end time, at which point it will become a VOD manifest.

- If a start parameter is specified without an end parameter, the resulting manifest will have a fixed start time corresponding to the specified start parameter, while the end of the manifest will grow as the live content progresses.

  **Note**
  
  For TS output, many playback devices start playback at the current time ("now"). To view the content from the actual start time of the playback window, viewers can seek back on the playback progress bar.

- If no start time or end time parameters are specified, the service will generate a standard manifest.
**Important**
When using time-shifted viewing, we recommend using consistent playback windows across player sessions, rather than generating a unique start or end time for each viewer. This yields better caching at the CDN, and will avoid running into potential throttling related to those requests, on the MediaPackage level.

**Time delay**

You can specify a duration (in seconds) for MediaPackage to delay when content is available to players. The minimum time is 0 seconds. The following rules determine the maximum time:

- If `startoverwindow` is equal to 0, the maximum time is 86,400 seconds (24 hours).
- If `startoverwindow` is not equal to 0, the maximum time is the value of `startoverwindow`.

Use `time_delay` to redefine the live point and make content available at a time that equals "now" minus the delay specified. With a 60-second `time_delay`, content that MediaPackage receives at 12:20 isn't available until 12:21. Requests for playback at 12:20 will be served with content from 12:19. Likewise, if you're serving content across time zones, you can set a `time_delay` equal to the time zone difference to make content available at, for example, 8:00 local time.

When you use `time_delay` in conjunction with a startover window, the time delay duration must be less than the startover window duration.

**Tip**
Use a `time_delay` to help reduce buffering during input switching when you're using input redundancy with short output segments. Note that the delay can increase latency in content playback.

- Query parameter notation – `time_delay` parameters are included at the end of the request URL

**Example time_delay**

```
https://cf98fa7b2ee4450e.mediapackagev2.us-east-1.amazonaws.com/out/v1/reference-streams/reference-channel/CMAF-endpoint/index-llhls.m3u8?time_delay=901
```

**Rules for start and end parameters**

Start and end parameters denote the beginning and end of a time-shifted manifest. The playback device can append parameters to the end of a manifest request or include the parameters within the request.

In all cases, the date and time must be notated in one of the following formats:

- ISO 8601 dates, such as 2017-08-18T21:18:54+00:00
- POSIX (or Epoch) time, such as 1503091134

The following topics describe the location rules by packager type.

**TS and CMAF parameter rules**

Start and end parameters in the URL request for TS content can use standard parameter notation.

- Query parameter notation – start and end parameters are included at the end of the request URL
Working with trick-play in AWS Elemental MediaPackage

Trick-play, sometimes called trick mode, provides a visual cue to viewers as they rewind, fast-forward, or seek through content in a digital video player. This helps the person using the video player to visualize where they are in the content timeline.

MediaPackage supports the following trick-play types:

**Supported trick-play types for live workflows**

<table>
<thead>
<tr>
<th>Streaming protocol</th>
<th>I-frame only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple HLS</td>
<td>✓</td>
</tr>
<tr>
<td>CMAF Apple HLS</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Topics**

- [Using I-frame playlists to enable trick-play (p. 52)](#)

### Using I-frame playlists to enable trick-play

MediaPackage supports live and on-demand trick-play by creating an I-frame playlist from an existing VOD asset or live stream. The I-frame playlist contains the I-frame only video segments that your player uses for the image thumbnails. For information about I-frame playlists, see the HTTP Live Streaming 2nd Edition specification: [https://datatracker.ietf.org/doc/html/rfc8216#section-4.3.3.6](https://datatracker.ietf.org/doc/html/rfc8216#section-4.3.3.6).

**To use an I-frame playlist to enable trick-play**

- In the MediaPackage console, choose **Include Iframe-only stream** when creating or editing an endpoint. MediaPackage generates I-frame only streams for each rendition in the parent playlist.

  This example TS parent playlist generated by MediaPackage uses EXT-X-I-FRAME-STREAM-INF tags for each rendition and includes the "index_1_iframe.m3u8" and "index_2_iframe.m3u8" child playlist URIs.

  ```
  #EXTM3U
  #EXT-X-VERSION:4
  ```
Working with data plane APIs in AWS Elemental MediaPackage

MediaPackage uses three data plane actions for ingesting and egressing video content. MediaPackage does not use these actions to configure Channel Group, Channel, or Origin Endpoint resources.

You can dynamically produce client manifests by appending parameters to the playback request. For more information, see Manifest filtering (p. 38).

**PutObject**

During ingest, uploads video segments from an encoder into a MediaPackage channel.

To call PutObject, perform an HTTP GET request to the ingest URL for a channel.

For more information about video format requirements for PutObject, see Supported input types (p. 3).

**GetObject**

Download video segments from an origin endpoint in MediaPackage.
To call `GetObject`, perform an HTTP GET request to an egress URL on an origin endpoint.

**GetHeadObject**

Retrieves just the HTTP headers of the video content.

To call `GetHeadObject`, perform an HTTP HEAD request to an egress URL on an origin endpoint.
Security in AWS Elemental MediaPackage

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that’s 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 MediaPackage, 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 MediaPackage. The following topics show you how to configure MediaPackage to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your MediaPackage resources.

**Topics**
- Data protection in AWS Elemental MediaPackage (p. 55)
- Identity and Access Management for AWS Elemental MediaPackage (p. 56)
- Compliance validation for AWS Elemental MediaPackage (p. 80)
- Resilience in AWS Elemental MediaPackage (p. 80)
- Infrastructure Security in AWS Elemental MediaPackage (p. 80)

Data protection in AWS Elemental MediaPackage

The AWS shared responsibility model applies to data protection in AWS Elemental MediaPackage. 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. This content includes 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.
• 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 140-2 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) 140-2.

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 MediaPackage 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.

Topics
• Implementing DRM with AWS Elemental MediaPackage (p. 56)

Implementing DRM with AWS Elemental MediaPackage

Use encryption to protect your content from unauthorized access. MediaPackage supports digital rights management (DRM). With DRM, you can make sure that once you distribute your content, only authorized viewers can watch it.

For information about using DRM with MediaPackage, see Content encryption and DRM in AWS Elemental MediaPackage (p. 34).

Identity and Access Management for AWS Elemental MediaPackage

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 MediaPackage resources. IAM is an AWS service that you can use with no additional charge.

Topics
• Audience (p. 57)
• Authenticating with identities (p. 57)
• Managing access using policies (p. 59)
• How AWS Elemental MediaPackage works with IAM (p. 61)
• Identity-based policy examples for MediaPackage (p. 66)
• Resource-based policy examples (p. 68)
• AWS managed policies for AWS Elemental MediaPackage (p. 75)
• Authenticating Requests (AWS Signature Version 4) (p. 77)
• Cross-service confused deputy prevention (p. 77)
Audience

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

**Service user** – If you use the MediaPackage service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more MediaPackage 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 MediaPackage, see Troubleshooting MediaPackage identity and access (p. 78).

**Service administrator** – If you’re in charge of MediaPackage resources at your company, you probably have full access to MediaPackage. It’s your job to determine which MediaPackage 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 MediaPackage, see How AWS Elemental MediaPackage works with IAM (p. 61).

**IAM administrator** – If you’re an IAM administrator, you might want to learn details about how you can write policies to manage access to MediaPackage. To view example MediaPackage identity-based policies that you can use in IAM, see Identity-based policy examples for MediaPackage (p. 66).

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 Signing AWS 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 Using multi-factor authentication (MFA) in AWS 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.
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 When to create an IAM user (instead of a role) in the IAM User Guide.

**IAM roles**

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

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

- **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 Creating a role for a third-party Identity Provider 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 [How IAM roles differ from resource-based policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/iam-role-vs-policy.html) 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.

• **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see [Actions, resources, and condition keys for AWS Elemental MediaPackage](https://docs.aws.amazon.com/mediapackage/latest/mediapackage-explorer.html) in the Service Authorization Reference.

• **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 [Creating a role to delegate permissions to an AWS service](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_rolesEK.html) 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 [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_rolesEK.html) in the IAM User Guide.

To learn whether to use IAM roles or IAM users, see [When to create an IAM role (instead of a user)](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_rolesEK.html) 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](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_policies_overview.html) 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 Creating IAM policies in the IAM User Guide.

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

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. 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 the 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 How SCPs work in the AWS Organizations User Guide.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies.
Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the IAM User Guide.

Multiple policy types

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

How AWS Elemental MediaPackage works with IAM

Before you use IAM to manage access to MediaPackage, learn what IAM features are available to use with MediaPackage.

IAM features you can use with MediaPackage

<table>
<thead>
<tr>
<th>IAM feature</th>
<th>MediaPackage support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity-based policies (p. 61)</td>
<td>Yes</td>
</tr>
<tr>
<td>Resource-based policies (p. 62)</td>
<td>Yes</td>
</tr>
<tr>
<td>Policy actions (p. 62)</td>
<td>Yes</td>
</tr>
<tr>
<td>Policy resources (p. 63)</td>
<td>Yes</td>
</tr>
<tr>
<td>Policy condition keys (service-specific) (p. 64)</td>
<td>Yes</td>
</tr>
<tr>
<td>ACLs (p. 64)</td>
<td>No</td>
</tr>
<tr>
<td>ABAC (tags in policies) (p. 64)</td>
<td>Yes</td>
</tr>
<tr>
<td>Temporary credentials (p. 65)</td>
<td>Yes</td>
</tr>
<tr>
<td>Principal permissions (p. 65)</td>
<td>Yes</td>
</tr>
<tr>
<td>Service roles (p. 65)</td>
<td>Yes</td>
</tr>
<tr>
<td>Service-linked roles (p. 66)</td>
<td>No</td>
</tr>
</tbody>
</table>

To get a high-level view of how MediaPackage 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 MediaPackage

| 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 Creating IAM 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 MediaPackage

To view examples of MediaPackage identity-based policies, see Identity-based policy examples for MediaPackage (p. 66).

Resource-based policies within MediaPackage

<table>
<thead>
<tr>
<th>Supports resource-based policies</th>
<th>Yes</th>
</tr>
</thead>
</table>

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 How IAM roles differ from resource-based policies in the IAM User Guide.

Policy actions for MediaPackage

<table>
<thead>
<tr>
<th>Supports policy actions</th>
<th>Yes</th>
</tr>
</thead>
</table>

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 MediaPackage actions, see Actions defined by AWS Elemental MediaPackage in the Service Authorization Reference.

Policy actions in MediaPackage use the following prefix before the action:

mediapackagev2

To specify multiple actions in a single statement, separate them with commas.
You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word Describe, include the following action:

```
"Action": "mediapackagev2:Describe*"
```

To view examples of MediaPackage identity-based policies, see Identity-based policy examples for MediaPackage (p. 66).

**Policy resources for MediaPackage**

<table>
<thead>
<tr>
<th>Supports policy resources</th>
<th>Yes</th>
</tr>
</thead>
</table>

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 Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as resource-level permissions.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

MediaPackage has the following resource ARNs:

```
arn:aws:mediapackagev2:us-east-1:111122223333:channelGroup/channelGroupName/
channelName
```

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

For example, to specify the 9a6b3953e242400eb805f324d95788e3 channel in your statement, use the following ARN:

```
"Resource": "arn:aws:mediapackagev2:us-east-1:111122223333:channelGroup/channelGroupName/channel/9a6b3953e242400eb805f324d95788e3"
```

To specify all instances that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:mediapackagev2:us-east-1:111122223333:channelGroup/channelGroupName/channel/**"
```

Some MediaPackage actions, such as those for creating resources, can't be performed on a specific resource. In those cases, you must use the wildcard (*).
"Resource": "*"

To see a list of MediaPackage resource types and their ARNs, see Resources defined by AWS Elemental MediaPackage in the Service Authorization Reference. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS Elemental MediaPackage.

To view examples of MediaPackage identity-based policies, see Identity-based policy examples for MediaPackage (p. 66).

Policy condition keys for MediaPackage

| 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.

To see a list of MediaPackage condition keys, see Condition keys for AWS Elemental MediaPackage in the Service Authorization Reference. To learn with which actions and resources you can use a condition key, see Actions defined by AWS Elemental MediaPackage.

To view examples of MediaPackage identity-based policies, see Identity-based policy examples for MediaPackage (p. 66).

ACLs in MediaPackage

| 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 MediaPackage

| Supports ABAC (tags in policies) | Yes |
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 What is ABAC? 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.

Using temporary credentials with MediaPackage

| 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 Switching to a 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 MediaPackage

| Supports principal permissions | Yes |

When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, resources, and condition keys for AWS Elemental MediaPackage in the Service Authorization Reference.

Service roles for MediaPackage

| Supports service roles | Yes |
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 Creating a role to delegate permissions to an AWS service in the IAM User Guide.

**Warning**
Changing the permissions for a service role might break MediaPackage functionality. Edit service roles only when MediaPackage provides guidance to do so.

### Choosing an IAM role in MediaPackage

When you create an asset resource in MediaPackage, you must choose a role to allow MediaPackage to access Amazon S3 on your behalf. If you previously created a service role or service-linked role, MediaPackage provides you with a list of roles to choose from. It’s important to choose a role that allows access to read from the Amazon S3 bucket and retrieve content.

### Service-linked roles for MediaPackage

| Supports service-linked roles | No |

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 service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a Yes in the Service-linked role column. Choose the Yes link to view the service-linked role documentation for that service.

### Identity-based policy examples for MediaPackage

By default, users and roles don’t have permission to create or modify MediaPackage 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 Creating IAM policies in the IAM User Guide.

For details about actions and resource types defined by MediaPackage, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Elemental MediaPackage in the Service Authorization Reference.

### Topics
- Policy best practices (p. 66)
- Using the MediaPackage console (p. 67)
- Allow users to view their own permissions (p. 67)

### Policy best practices

Identity-based policies determine whether someone can create, access, or delete MediaPackage 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 IAM Access Analyzer policy validation 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 Configuring MFA-protected API access 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 MediaPackage console

To access the AWS Elemental MediaPackage console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the MediaPackage 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 MediaPackage console, also attach the MediaPackage ReadOnly AWS managed policy to the entities. For more information, see Adding permissions to a user in the IAM User Guide.

AWS Elemental MediaPackage ReadOnly

### 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.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "ViewOwnUserInfo",
         "Effect": "Allow",
```
Resource-based policy examples

A resource policy is an access policy option available for granting permission to your MediaPackage resources. Resource-based policies are JSON policy documents.

The topics in this section describe the key policy language elements, with focus on MediaPackage–specific details, and provide example resource policies. We recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your MediaPackage resources.

To learn how to attach a resource-based policy to a channel, see Creating a channel (p. 23).

Topics
- Policies and Permissions in MediaPackage (p. 68)
- Ingest authorization (p. 71)
- Origin endpoint authorization (p. 73)

Policies and Permissions in MediaPackage

This page provides an overview of resource policies in MediaPackage and describes the basic elements of a policy. Each listed element links to more details about that element and examples of how to use it.

For a complete list of MediaPackage actions, resources, and conditions, see Actions, resources, and condition keys for AWS Elemental MediaPackage in the AWS General Reference.

In its most basic sense, a policy contains the following elements:

- **Resources** - Channels and origin endpoints are the MediaPackage resources for which you can allow or deny permissions. In a policy, you use the Amazon Resource Name (ARN) to identify the resource. For more information, see MediaPackage resources (p. 71).
• **Actions** - For each resource, MediaPackage supports a set of operations. You identify resource operations that you will allow (or deny) by using action keywords. For more information, see [IAM JSON Policy Elements: Action](#).

• **Effect** - This determines what the effect will be when the user requests the specific action. This can be either *allow* or *deny*.

If you do not explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource. You might do this to make sure that a user can't access the resource, even if a different policy grants access. For more information, see [IAM JSON Policy Elements: Effect](#).

• **Principal** - The account or user who is allowed access to the actions and resources in the statement. In a resource policy, the principal is the user, account, service, or other entity that is the recipient of this permission. For more information, see [Principals](#) and [AWS JSON Policy Elements: Principal](#).

• **Condition** - These are the conditions for when a policy is in effect. You can use AWS-wide keys and MediaPackage-specific keys to specify conditions in an MediaPackage access policy. For more information, see [IAM JSON Policy Elements: Condition](#).

To illustrate, consider the following Allow policy. With this policy in effect, Jane Doe has mediapackagev2:GetObject and mediapackagev2:GetHeadObject permissions on all objects from the specified origin endpoint under the condition that the request are made over HTTPS.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowJaneDoe",
         "Effect": "Allow",
         "Principal": { "AWS": "arn:aws:iam::accountID:user/JaneDoe" },
         "Action": ["mediapackagev2:GetObject","mediapackagev2:GetHeadObject"],
         "Resource": "arn:aws:mediapackagev2:Region:AccountID:channelGroup/ChannelGroupName/channel/ChannelName/originEndpoint/OriginEndpointName",
         "Condition": {
            "Bool": { "aws:SecureTransport": "true" }
         }
      }
   ]
}
```

Resource policies are specific to the resources to which they are applied. Applying a policy to a particular origin endpoint that allows anonymous GetObject doesn't automatically apply GetObject to other endpoints even if the ARN matches. For instance, if you apply a policy to origin endpoint abcd01234567890, it only applies to that endpoint and not to another endpoint with a similar ARN, like 021345abcdef67890. This means that the policy is not automatically applied to any other resource with a matching ARN, and you must apply the policy explicitly to each resource that requires it.

For more, see the topics below. For complete policy language information, see [Policies and Permissions](#) and [IAM JSON Policy Reference](#) in the [IAM User Guide](#).

**Topics**

- [Principals](#)
- [Actions, resources, and condition keys in MediaPackage](#)

**Principals**

The **Principal** element specifies the user, account, service, or other entity that is allowed or denied access to a resource. For more information, see [Principal](#) in the [IAM User Guide](#).
Grant permissions to an AWS account

To grant permissions to an AWS account, identify the account using the following format.

"AWS": "account-ARN"

The following are examples.

"Principal": {"AWS": "arn:aws:iam::AccountIDWithoutHyphens:root"}


Grant permissions to an IAM user

To grant permission to an IAM user within your account, you must provide an "AWS": "user-ARN" name-value pair.

"Principal": {"AWS": "arn:aws:iam::account-number-without-hyphens:user/username"}

Note

If an IAM identity is deleted after you update your resource policy, the resource policy will show a unique identifier in the principal element instead of an ARN. These unique IDs are never reused, so you can safely remove principals with unique identifiers from all of your policy statements. For more information about unique identifiers, see IAM identifiers in the IAM User Guide.

Grant anonymous permissions

To grant permission to everyone, also referred as anonymous access, you set the wildcard ("*") as the Principal value. For example, if you want to use clients with no AWS authorization to their origin endpoints.

"Principal": "*

"Principal": {"AWS": "*

Using "Principal": "*" with an Allow effect in a resource-based policy allows anyone, even if they're not signed in to AWS, to access your resource.

Using "Principal": { "AWS": "*" } with an Allow effect in a resource-based policy allows any root user, IAM user, assumed-role session, or federated user in any account in the same partition to access your resource.

For anonymous users, these two methods are equivalent. For more information, see All principals in the IAM User Guide.

You cannot use a wildcard to match part of a principal name or ARN.

Important

Because anyone can create an AWS account, the security level of these two methods is equivalent, even though they function differently.

Warning

Use caution when granting anonymous access to your MediaPackage origin endpoints. When you grant anonymous access, anyone in the world can access your bucket. We highly recommend that you never grant any kind of anonymous write access to your origin endpoints.
Actions, resources, and condition keys in MediaPackage

AWS Elemental MediaPackage (service prefix: mediapackagev2) provides service-specific resources, actions, and condition context keys for use in IAM permission policies. For the full list, see Actions, resources, and condition keys for AWS Elemental MediaPackage in the AWS General Reference.

MediaPackage Actions

MediaPackage defines a set of permissions that you can specify in a policy. These are keywords, each of which maps to a specific MediaPackage operation. When you use an action in a policy, you usually allow or deny access to the API operation or CLI command with the same name. However, in some cases, a single action controls access to more than one operation. Alternatively, some operations require several different actions.

MediaPackage resources

The following common Amazon Resource Name (ARN) format identifies resources in AWS:

```
arn:${Partition}:mediapackagev2:${Region}:${AccountID}:channelGroup/${ChannelGroupName}/channel/${ChannelName}/originEndpoint/${OriginEndpointName}
```

For information about ARNs, see Amazon Resource Names (ARNs) in the AWS General Reference.

For information about resources, see IAM JSON Policy Elements: Resource in the IAM User Guide.

A MediaPackage ARN includes the following:

- **Partition** - aws is a common partition name. If your resources are in the China (Beijing) Region, aws-cn is the partition name.
- **Region** - The AWS Region.
- **AccountID** - Your AWS account number.
- **ChannelGroupName** - The name of the channel group.
- **ChannelName** - The name of the channel.
- **OriginEndpointName** - The name of the origin endpoint.

MediaPackage Conditions keys

The access policy language enables you to specify conditions when granting permissions. To specify conditions for when a policy is in effect, you can use the optional Condition element, or Condition block, to specify conditions for when a policy is in effect. You can use predefined AWS-wide keys and MediaPackage-specific keys to specify conditions in an MediaPackage access policy. In the Condition element, you build expressions in which you use Boolean operators (equal, less than, etc.) to match your condition against values in the request.

Ingest authorization

MediaPackage ingest requests usually originate from a video encoder.

Topics

- AWS Elemental MediaLive (p. 71)
- AWS Elemental Live (p. 72)
- Third-party encoders (p. 73)

AWS Elemental MediaLive

This example illustrates a channel policy that permits MediaLive to ingest MediaPackage.
AWS Elemental MediaPackage AWS
Elemental MediaPackage User Guide
Resource-based policy examples

```json
{
    "Version": "2012-10-17",
    "Id": "AllowMediaLiveChannelToIngestToEmpChannel",
    "Statement": [
        {
            "Sid": "AllowMediaLiveRoleToAccessEmpChannel",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountID:role/MediaLiveAccessRole"
            },
            "Action": "mediapackagev2:PutObject",
            "Resource": "arn:aws:mediapackagev2:Region:AccountID:channelGroup/ChannelGroupName/channel/ChannelName"
        }
    ]
}
```

**AWS Elemental Live**

If you provide Elemental Live with an access key ID and secret access key, it can request access as an IAM identity. To grant your Elemental Live encoder access to your MediaPackage channel, you can apply the following `Allow` policy.

1. In IAM, create an IAM user such as `ElementalLiveMediaPackageUser` with **Programmatic access**.
2. In MediaPackage, create or edit a channel to include the following channel policy.

```json
{
    "Version": "2012-10-17",
    "Id": "AllowIamUser",
    "Statement": [
        {
            "Sid": "AllowIamUserToEmpChannel",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountID:user/ElementalLiveMediaPackageUser"
            },
            "Action": "mediapackagev2:PutObject",
            "Resource": "arn:aws:mediapackagev2:Region:AccountID:channelGroup/ChannelGroupName/channel/ChannelName"
        }
    ]
}
```

3. In IAM, create an access key for `ElementalLiveMediaPackageAccessUser`. Save the access key .csv file in a secure location to retain a permanent record of the access key ID and secret access key.

   The access key ID looks like this: AKIAIOSFODNN7EXAMPLE

   The secret access key looks like this: wJalrXUttnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

   For more information, see Programmatic access in the AWS General Reference.

4. Share the access key ID and the secret access key with the Elemental Live operator. Do not give the username and password to the operator.

By following these steps, you'll create an AWS user with the necessary permissions required to allow Elemental Live to make requests to MediaPackage. When the operator sets up the output with MediaPackage as the destination, they will enter the access key ID and secret access key. During the Elemental Live event, Elemental Live sends these two IDs to the AWS service instead of the username
and password, providing authorization to AWS for the Elemental Live node to make requests to MediaPackage.

**Third-party encoders**

Third-party encoders that support AWS authorization operate similarly to Elemental Live, as described earlier. To grant access, create an IAM user and a MediaPackage channel resource policy that permits the user to call `PutObject`. On the encoder's side, use the IAM user access key ID and secret access key to sign the requests.

**Origin endpoint authorization**

MediaPackage egress requests usually originate from CDNs, but they may also come from other sources such as customer-owned monitoring scripts or operators using web browsers like Safari or Chrome to view the video stream and identify any issues.

**Topics**

- Third-party CDNs that support AWS authorization (p. 73)
- Clients that don't support AWS authorization (p. 74)

**Third-party CDNs that support AWS authorization**

To authorize an external CDN that supports AWS authorization, you need to create a specific IAM user for the CDN, allow access in their origin endpoint policy, and provide the CDN with the AWS access key ID and secret access key for the IAM user. For example, if you want to give your CDN provider access to your MediaPackage origin endpoint, you can follow the following procedure.

1. In IAM, create an IAM user such as `CDNProviderMediaPackageAccessUser` with **Programmatic access**.
2. In MediaPackage, create or edit an origin endpoint to include the following endpoint policy.

   ```json
   {
     "Version": "2012-10-17",
     "Id": "PolicyForCDNProviderPrivateContent",
     "Statement": [
       {
         "Sid": "AllowCDNProviderUser",
         "Effect": "Allow",
         "Principal": { "AWS": "arn:aws:iam::AccountID:user/CDNProviderMediaPackageAccessUser" },
         "Action": "mediapackagev2:GetObject",
         "Resource": "arn:aws:mediapackagev2:Region:AccountID:channelGroup/ChannelGroupName/channel/ChannelName/originEndpoint/OriginEndpointName"
       }
     ]
   }
   ```

3. In IAM, create an access key for `CDNProviderMediaPackageAccessUser`. Save the access key .csv file in a secure location to retain a permanent record of the access key ID and secret access key.

   The access key ID looks like this: AKIAIOSFODNN7EXAMPLE

   The secret access key looks like this: wJalrXUttnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

   For more information, see **Programmatic access** in the **AWS General Reference**.

4. Follow the instructions in your CDN provider's documentation for authenticating with AWS access keys.
By following these steps, you'll create an AWS user with the necessary permissions required to allow the external CDN make requests to MediaPackage. When the CDN provider sets up the output with MediaPackage as the destination, they will enter the access key ID and secret access key. During the event, the provider sends these two IDs to the AWS service instead of the username and password, providing authorization to make requests to MediaPackage.

**Clients that don't support AWS authorization**

Clients without AWS authorization support can be granted access to origin endpoints either by enabling anonymous access or by restricting access to specific IP ranges using the `aws:SourceIp` condition key. This is useful for clients such as external CDNs that don't support AWS authorization, as well as monitoring scripts and human operators who may use web browsers to visually inspect a video stream. For information about condition keys, see [IAM JSON Policy Elements: Condition](#).

**Anonymous access**

Consider the following `Allow` policy. With this policy in effect, MediaPackage allows anonymous access to the `mediapackagev2:GetObject` action on the channel resource in the policy.

```json
{
   "Version": "2012-10-17",
   "Id": "AnonymousAccessPolicy",
   "Statement": [
   {
      "Sid": "AllowAnonymousAccess",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "mediapackagev2:GetObject",
      "Resource": "arn:aws:mediapackagev2:Region:AccountID:channelGroup/ChannelGroupName/channel/ChannelName/originEndpoint/OriginEndpointName"
   }
   ]
}
```

MediaPackage doesn't support anonymous access for `PutObject` API calls.

**Cross-account access**

Consider the following `Allow` policy. With this policy in effect, MediaPackage allows, across accounts (accountID and differentAccountID), the `mediapackagev2:GetObject` action on the channel resource in the policy.

```json
{
   "Version": "2012-10-17",
   "Statement": [
   {
      "Sid": "AllowCrossAccountAccess",
      "Effect": "Allow",
      "Principal": {
         "AWS": "arn:aws:iam::differentAccountID:root"},
      "Action": "mediapackagev2:GetObject",
      "Resource": "arn:aws:mediapackagev2:Region:AccountID:channelGroup/ChannelGroupName/channel/ChannelName"
   }
   ]
}
```

**Restrict access by IP range**

Consider the following `Allow` policy. With this policy in effect, MediaPackage restricts access to IP addresses in the range 203.0.113.0 to 203.0.113.255 using the `aws:SourceIp` condition key. For information about condition keys, see [IAM JSON Policy Elements: Condition](#).
AWS managed policies for AWS Elemental MediaPackage

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they’re available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the IAM User Guide.

AWS managed policy: AWSElementalMediaPackageV2FullAccess

This policy grants contributor permissions that allow all actions on all live resources in MediaPackage.

You can attach the AWSElementalMediaPackageV2FullAccess policy to your IAM identities.
AWS managed policy: AWSElementalMediaPackageV2ReadOnly

This policy grants contributor permissions that allow read-only actions on all live resources in MediaPackage.

You can attach the AWSElementalMediaPackageV2ReadOnly policy to your IAM identities.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "mediapackagev2:List*",
      "mediapackagev2:Get**
    ],
    "Resource": "*"
  }
}
```

MediaPackage updates to AWS managed policies

View details about updates to AWS managed policies for MediaPackage since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the MediaPackage Document history (p. 103) page.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWSElementalMediaPackageV2ReadOnly – New policy</td>
<td>MediaPackage added a new full-access policy for live resources. This policy allows all actions on all live resources in MediaPackage.</td>
<td>July 25, 2023</td>
</tr>
<tr>
<td>AWSElementalMediaPackageV2ReadOnly – New policy</td>
<td>MediaPackage added a new read-only policy for live resources. This policy allows read-only actions on all live resources in MediaPackage.</td>
<td>July 25, 2023</td>
</tr>
<tr>
<td>MediaPackage started tracking changes</td>
<td>MediaPackage started tracking changes for its AWS managed policies.</td>
<td>July 25, 2023</td>
</tr>
</tbody>
</table>
Authenticating Requests (AWS Signature Version 4)

Every interaction with MediaPackage is either authenticated or anonymous. This section explains request authentication with the AWS Signature Version 4 algorithm.

**Note**
If you use the AWS SDKs or AWS CLI to send your requests, you don’t need to read this section because these tools authenticate your requests by using access keys that you provide. You must only sign AWS API requests as described in this documentation if you do not use an AWS SDK or AWS CLI to send AWS API requests.

When you send API requests to AWS, you must sign them so that AWS can identify the sender. For security, most requests are signed using your AWS security credentials.

When MediaPackage receives an authenticated request, it recreates the signature using the authentication information contained in the request. If the signatures match, MediaPackage processes the request. Otherwise, it rejects the request.

AWS Signature Version 4 is the AWS signing protocol. AWS also supports an extension, Signature Version 4A, which supports signatures for multi-Region API requests.

For additional information about AWS Signature Version 4, see:
- Signing AWS API requests in the IAM User Guide
- Authenticating Requests (AWS Signature Version 4) in the Amazon Simple Storage Service API Reference

Creating a signed AWS API request

For steps to create a signed AWS API request, see:
- Create a signed AWS API request in the IAM User Guide
- Signature Calculations for the Authorization Headers: Transferring Payload in a Single Chunk in the Amazon Simple Storage Service API Reference

Troubleshooting signed AWS API requests

For troubleshooting help with your signed requests, see Troubleshoot signed requests for AWS APIs in the IAM User Guide.

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 MediaPackage gives another service to the resource. Use aws:SourceArn if you want only one resource to be associated with the cross-service access. Use aws:SourceAccount if you want to allow any resource in that account to be associated with the cross-service use.
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 wildcard characters (*) for the unknown portions of the ARN. For example,

```
arn:aws:servicename:::123456789012:
```

If the `aws:SourceArn` value does not contain the account ID, such as an Amazon S3 bucket ARN, you must use both global condition context keys to limit permissions.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in MediaPackage to prevent the confused deputy problem.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "mediapackage.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:mediapackage:*:123456789012:harvest_jobs/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

**Troubleshooting MediaPackage identity and access**

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

**Topics**

- I'm not authorized to perform an action in MediaPackage (p. 78)
- I'm not authorized to perform `iam:PassRole` (p. 79)
- I want to allow people outside of my AWS account to access my MediaPackage resources (p. 79)

**I'm not authorized to perform an action in MediaPackage**

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 `mediapackagev2:GetWidget` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
mediapackagev2: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 `mediapackagev2:GetWidget` action.
If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I'm 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 MediaPackage.

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 MediaPackage. 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 MediaPackage 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 MediaPackage supports these features, see How AWS Elemental MediaPackage works with IAM (p. 61).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the IAM User Guide.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the IAM User Guide.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the IAM User Guide.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the IAM User Guide.

Learn More

For more information about identity and access management for MediaPackage, continue to the following pages:

- How AWS Elemental MediaPackage works with IAM (p. 61)
- Identity-based policy examples for MediaPackage (p. 66)
- Cross-service confused deputy prevention (p. 77)
- Troubleshooting MediaPackage identity and access (p. 78)
Compliance validation for AWS Elemental MediaPackage

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 and Compliance Quick Start Guides** – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- **Architecting for HIPAA Security and Compliance on Amazon Web Services** – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.
  
  **Note**
  Not all AWS services are HIPAA eligible. For more information, see the HIPAA Eligible Services Reference.

- **AWS Compliance Resources** – This collection of workbooks and guides might apply to your industry and location.
- **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.
- **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 MediaPackage

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 Availability 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.

Infrastructure Security in AWS Elemental MediaPackage

As a managed service, AWS Elemental MediaPackage 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 MediaPackage 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.
Logging and monitoring in MediaPackage

Monitoring is an important part of maintaining the reliability, availability, and performance of MediaPackage and your other AWS solutions. AWS provides the following monitoring tools to watch MediaPackage, 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](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/index.html).

- **Amazon CloudWatch Events** delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events enables automated event-driven computing, as you can write rules that watch for certain events and trigger automated actions in other AWS services when these events happen. For more information, see the [Amazon CloudWatch Events User Guide](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatchEventsOverview.html).

- **AWS CloudTrail** captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the [AWS CloudTrail User Guide](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/AWS-CloudTrail.html).

Topics
- Monitoring MediaPackage with Amazon CloudWatch metrics (p. 82)
- Logging AWS Elemental MediaPackage API calls with AWS CloudTrail (p. 91)
- Monitoring manifest update time (p. 93)
- MediaPackage response headers (p. 95)

Monitoring MediaPackage with Amazon CloudWatch metrics

You can monitor MediaPackage using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the [Amazon CloudWatch User Guide](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/index.html).

To view metrics using the MediaPackage console

MediaPackage displays metrics throughout the console.

2. Navigate to the appropriate page to view metrics:
   - For metrics on all channel groups, go to the **Channel groups** page.
   - For metrics on all channels and origin endpoints associated with your channel group in the AWS Region, go to the channel group's details page.
   - For metrics on a specific channel and all of its origin endpoints, go to the channel's details page.
   - For metrics on a specific origin endpoint, go to the origin endpoint's details page.
3. (Optional) To refine the metrics view, choose **Open in CloudWatch**.

**To view metrics using the CloudWatch console**

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

2. In the navigation pane, choose **Metrics**.
3. Under **All metrics**, choose the **AWS/MediaPackage** namespace.
4. Choose the metric dimension to view the metrics (for example, choose channel to view metrics per channel).

**To view metrics using the AWS CLI**

At a command prompt, enter the following command:

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

**Topics**

- [MediaPackage live content metrics (p. 83)](#)

**MediaPackage live content metrics**

The AWS/MediaPackage namespace includes the following metrics for live content. MediaPackage publishes metrics to CloudWatch every minute, if not sooner.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ActiveInput</strong></td>
<td>Indicates if an input has been used as the source for an endpoint in MediaPackage (it has been active). A value of 1 indicates that the input was active, and a 0 (zero) indicates that it wasn't. Units: None</td>
</tr>
<tr>
<td></td>
<td>Valid dimension:</td>
</tr>
<tr>
<td></td>
<td>• Combination of IngestEndpoint and OriginEndpoint</td>
</tr>
<tr>
<td><strong>EgressBytes</strong></td>
<td>Number of bytes that MediaPackage successfully sends for each request. If MediaPackage doesn't receive any requests for output in the specified interval, then no data is given. Units: Bytes</td>
</tr>
<tr>
<td></td>
<td>Valid statistics:</td>
</tr>
<tr>
<td></td>
<td>• Average – Average bytes (Sum/SampleCount) that MediaPackage outputs over the configured interval.</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>• Maximum</td>
<td>Largest individual output request (in bytes) made to MediaPackage.</td>
</tr>
<tr>
<td>• Minimum</td>
<td>Smallest individual output request (in bytes) made to MediaPackage.</td>
</tr>
<tr>
<td>• SampleCount</td>
<td>Number of requests that's used in the statistical calculation.</td>
</tr>
<tr>
<td>• Sum</td>
<td>Total number of bytes that MediaPackage outputs over the configured interval.</td>
</tr>
</tbody>
</table>

Valid dimensions:

• ChannelGroup
• RequestType
• Combination of ChannelGroup and Channel
• Combination of ChannelGroup, Channel, and RequestType
• Combination of ChannelGroup, Channel, OriginEndpoint
• Combination of ChannelGroup, Channel, OriginEndpoint, and RequestType
• No dimension
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EgressRequestCount</td>
<td>Number of content requests that MediaPackage receives. If MediaPackage doesn't receive any requests for output in the specified interval, then no data is given.</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td></td>
<td>Valid statistics:</td>
</tr>
<tr>
<td></td>
<td>• Sum – Total number of output requests that MediaPackage receives.</td>
</tr>
<tr>
<td></td>
<td>Valid dimensions:</td>
</tr>
<tr>
<td></td>
<td>• ChannelGroup</td>
</tr>
<tr>
<td></td>
<td>• RequestType</td>
</tr>
<tr>
<td></td>
<td>• StatusCodeRange</td>
</tr>
<tr>
<td></td>
<td>• OutputType and StatusCodeRange</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup and RequestType</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup and StatusCodeRange</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup, RequestType, and StatusCodeRange</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup and Channel</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup, Channel, and RequestType</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup, Channel, and OriginEndpoint</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup, Channel, and StatusCodeRange</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup, Channel, OriginEndpoint, and RequestType</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup, Channel, RequestType, and StatusCodeRange</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup, Channel, OriginEndpoint, and StatusCodeRange</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup, Channel, OriginEndpoint, RequestType, and StatusCodeRange</td>
</tr>
<tr>
<td></td>
<td>• No dimension</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EgressResponseTime</td>
<td>The time that it takes MediaPackage to process each output request. If MediaPackage doesn't receive any requests for output in the specified interval, then no data is given. Units: Milliseconds</td>
</tr>
</tbody>
</table>

Valid statistics:
- **Average** – Average amount of time (Sum/SampleCount) that it takes MediaPackage to process output requests over the configured interval.
- **Maximum** – Longest amount of time (in milliseconds) that it takes MediaPackage to process an output request and provide a response.
- **Minimum** – Shortest amount of time (in milliseconds) that it takes MediaPackage to process an output request and provide a response.
- **SampleCount** – Number of requests that's used in the statistical calculation.
- **Sum** – Total amount of time that it takes MediaPackage to process output requests over the configured interval.

Valid dimensions:
- ChannelGroup
- RequestType
- Combination of ChannelGroup and Channel
- Combination of ChannelGroup, Channel, and RequestType
- Combination of ChannelGroup, Channel, and OriginEndpoint
- Combination of ChannelGroup, Channel, OriginEndpoint, and RequestType
- No dimension
## Metric Description

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IngressBytes</td>
<td>Number of bytes of content that MediaPackage receives for each input request. If MediaPackage doesn't receive any requests for input in the specified interval, then no data is given.</td>
</tr>
</tbody>
</table>

**Units:** Bytes

**Valid statistics:**

- **Average** – Average bytes (Sum/SampleCount) that MediaPackage receives over the configured interval.
- **Maximum** – Largest individual input request (in bytes) made to MediaPackage.
- **Minimum** – Smallest individual input request (in bytes) made to MediaPackage.
- **SampleCount** – Number of requests that's used in the statistical calculation.
- **Sum** – Total number of bytes that MediaPackage receives over the configured interval.

**Valid dimensions:**

- **ChannelGroup**
- **Combination of ChannelGroup and Channel**
- **Combination of ChannelGroup, Channel, and IngestEndpoint**
- **No dimension**
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
</table>
| IngressRequestCount | Number of input requests that MediaPackage receives. If MediaPackage doesn't receive any requests for input in the specified interval, then no data is given.  
Units: Count  
Valid statistics:  
  • Sum – Total number of input manifest requests that MediaPackage receives.  
Valid dimensions:  
  • ChannelGroup  
  • StatusCodeRange  
  • Combination of ChannelGroup and StatusCodeRange  
  • Combination of ChannelGroup and Channel  
  • Combination of ChannelGroup, Channel, and IngestEndpoint  
  • Combination of ChannelGroup, Channel, and StatusCodeRange  
  • Combination of ChannelGroup, Channel, IngestEndpoint, and StatusCodeRange  
  • No dimension |
Live content metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IngressResponseTime</td>
<td>The time that it takes MediaPackage to process each input request. If MediaPackage doesn't receive any requests for input in the specified interval, then no data is given.</td>
</tr>
<tr>
<td></td>
<td>Units: Milliseconds</td>
</tr>
<tr>
<td></td>
<td>Valid statistics:</td>
</tr>
<tr>
<td></td>
<td>• Average – Average amount of time (Sum/SampleCount) that it takes MediaPackage to process input requests over the configured interval.</td>
</tr>
<tr>
<td></td>
<td>• Maximum – Longest amount of time (in milliseconds) that it takes MediaPackage to process an input request and provide a response.</td>
</tr>
<tr>
<td></td>
<td>• Minimum – Shortest amount of time (in milliseconds) that it takes MediaPackage to process an input request and provide a response.</td>
</tr>
<tr>
<td></td>
<td>• SampleCount – Number of requests that's used in the statistical calculation.</td>
</tr>
<tr>
<td></td>
<td>• Sum – Total amount of time that it takes MediaPackage to process input requests over the configured interval.</td>
</tr>
<tr>
<td></td>
<td>Valid dimensions:</td>
</tr>
<tr>
<td></td>
<td>• ChannelGroup</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup and Channel</td>
</tr>
<tr>
<td></td>
<td>• Combination of ChannelGroup, Channel, and IngestEndpoint</td>
</tr>
<tr>
<td></td>
<td>• No dimension</td>
</tr>
</tbody>
</table>

**MediaPackage live dimensions**

You can filter the AWS/MediaPackage data using the following dimensions.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Dimension</td>
<td>Metrics are aggregated and shown for all channels, endpoints, or status codes.</td>
</tr>
<tr>
<td>Channel</td>
<td>Metrics are shown only for the specified channel.</td>
</tr>
<tr>
<td></td>
<td>Value: The autogenerated GUID of the channel.</td>
</tr>
<tr>
<td></td>
<td>Can be used alone or with other dimensions:</td>
</tr>
<tr>
<td></td>
<td>• Alone to show metrics for only the specified channel.</td>
</tr>
</tbody>
</table>
### Dimension | Description
--- | ---
IngestEndpoint | Metrics are shown only for the specified ingest endpoint on a channel.  
Value: The autogenerated GUID of the ingest endpoint.  
Can be used with the following dimensions:  
- With the channel dimension to show metrics for the specified ingest endpoint that's associated with the specified channel.  
- With the originEndpoint dimension to show metrics for the specified ingest endpoint that's associated with the specified endpoint.

OriginEndpoint | Metrics are shown for the specified channel and endpoint combination.  
Value: The autogenerated GUID of the endpoint.  
Must be used with the channel dimension.

StatusCodeRange | Metrics are shown for the specified status code range.  
Value: 2xx, 3xx, 4xx, or 5xx.  
Can be used alone or with other dimensions:  
- Alone to show all output requests for the specified status range.  
- With the channel dimension to show output requests for all endpoints that are associated with the specified channel, with the specified status code range.  
- With the channel and originEndpoint dimensions to show output requests with a specific status code range on the specified endpoint that's associated with the specified channel.
### Dimension | Description
--- | ---
**ChannelGroup** | Metrics are shown only for the specified channel group.  
Value: The name of the channel group.  
Can be used alone or with other dimensions:  
- Alone to show metrics only for the specified channel group.  
- With the channel dimension to show metrics for the specified channel that's associated with the specified channel group.  
- With the statusCodeRange dimension to show metrics for the specified status code ranges that are associated with the specified channel group.

**RequestType** | Metrics are shown only for the specified request type.  
Value: Either manifest or segment, signifying the type of content being filtered in the metric.  
Can be used alone or with other dimensions:  
- Alone to show metrics only for the specified request type.  
- With the statusCodeRange dimension to show metrics for the specified status code range associated with the specified request type.  
- With the originEndpoint dimension to show metrics for the specified request type that's associated with the specified origin endpoint.

---

## Logging AWS Elemental MediaPackage API calls with AWS CloudTrail

Logging is available with only live workflows in MediaPackage.

MediaPackage is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in MediaPackage. CloudTrail captures all API calls for MediaPackage as events. These include calls from the MediaPackage console and code calls to the MediaPackage API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for MediaPackage. 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 MediaPackage, 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](https://docs.aws.amazon.com/AmazonCloudTrail/latest/userguide/what-is-cloudtrail.html).
MediaPackage information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in MediaPackage, 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 account. For more information, see Viewing events with CloudTrail event history.

For an ongoing record of events in your account, including events for MediaPackage, 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 AWS Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All MediaPackage actions are logged by CloudTrail and are documented in the MediaPackage Live API reference. For example, calls to the CreateChannel, CreateOriginEndpoint, and RotateIngestEndpointCredentials operations generate entries in the CloudTrail log files.

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

- Whether the request was made with root user or IAM user credentials
- Whether the request was made with temporary security credentials for a role or federated user
- Whether the request was made by another AWS service

For more information, see the CloudTrail userIdentity element.

Understanding MediaPackage 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 UpdateChannel operation:

```json
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "ABCDEFGHIJKL123456789",
        "arn": "arn:aws:sts::444455556666:assumed-role/Admin/testUser",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2023-03-17T00:50:58Z"
            }
        }
    }
}
```
Monitoring manifest update time

MediaPackage playback responses include the following custom headers that indicate when MediaPackage last modified the manifest in non-dynamic ad insertion workflows. These headers are helpful when troubleshooting issues related to stale manifests.

**X-MediaPackage-Manifest-Last-Sequence**

This is the highest segment sequence number in the manifest.
X-MediaPackage-Manifest-Last-Updated

The epoch timestamp in milliseconds when MediaPackage generates the segment referred to in X-MediaPackage-Manifest-Last-Sequence.

Manifest examples

HLS manifest

MediaPackage determines the X-MediaPackage-Manifest-Last-Sequence value from the last segment in the manifest. For example, in the following manifest index_1_3.ts is the highest segment sequence number, so the value of X-MediaPackage-Manifest-Last-Sequence is 3. The value of X-MediaPackage-Manifest-Last-Updated corresponds to the epoch timestamp in milliseconds when MediaPackage generates the last segment in the manifest.

```xml
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:8
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:7.500,
index_1_0.ts?m=1583172400
#EXTINF:7.500,
index_1_1.ts?m=1583172400
#EXTINF:7.500,
index_1_2.ts?m=1583172400
#EXTINF:7.500,
index_1_3.ts?m=1583172400
#EXT-X-ENDLIST
```

CMAF manifest

Similar to HLS, MediaPackage determines the X-MediaPackage-Manifest-Last-Sequence value from the last segment in the manifest. For example, in the following manifest ../cmafseg_video_1_10.mp4?m=1621616399 is the highest segment sequence number, so the value of X-MediaPackage-Manifest-Last-Sequence is 10. The value of X-MediaPackage-Manifest-Last-Updated corresponds to the epoch timestamp in milliseconds when MediaPackage generates the last segment in the manifest.

```xml
#EXTM3U
#EXT-X-VERSION:6
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-TARGETDURATION:12
#EXT-X-MEDIA-SEQUENCE:1
#EXT-X-MAP:URI="../cmafseg_video_1_track_1098178399_csid_aaa_2_init.mp4"
#EXTINF:11.250,
../cmafseg_video_1_1.mp4?m=1621616399
#EXTINF:11.250,
../cmafseg_video_1_2.mp4?m=1621616399
#EXTINF:11.250,
../cmafseg_video_1_3.mp4?m=1621616399
#EXTINF:11.250,
../cmafseg_video_1_4.mp4?m=1621616399
```
MediaPackage response headers

Use the following AWS Elemental MediaPackage response headers to help you build your workflows. For more information about response headers associated with manifest monitoring, see Monitoring manifest update time.

<table>
<thead>
<tr>
<th>Header name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Amzn-Mediapackage-Active-Input</td>
<td>1 or 2</td>
<td>The active input pipeline when MediaPackage serves a given media segment.</td>
</tr>
<tr>
<td>X-Amzn-Mediapackage-Channel-Id</td>
<td>The channel-name value in the API.</td>
<td>Use separately from, or in addition to, X-Amzn-Mediapackage-Channel-UniqueId, to identify a given channel in the CDN logs. Channel names are unique only within a given region.</td>
</tr>
<tr>
<td>X-Amzn-Mediapackage-Channel-UniqueId</td>
<td>The unique identifier of the channel.</td>
<td>Use separately from, or in addition to, X-Amzn-Mediapackage-Channel-Id, to identify a given channel in the CDN logs. Channel names are unique only within a given region. Using X-Amzn-Mediapackage-Channel-UniqueId is also helpful for support requests.</td>
</tr>
<tr>
<td>X-Amzn-Mediapackage-Endpoint-Id</td>
<td>The manifest-name value in the API.</td>
<td>Use separately from, or in addition to, X-Amzn-Mediapackage-Endpoint-UniqueId, to identify a given endpoint in the CDN logs. Endpoint names are unique only within a given channel and region.</td>
</tr>
<tr>
<td>X-Amzn-Mediapackage-Endpoint-UniqueId</td>
<td>The unique identifier of the endpoint.</td>
<td>Use separately from, or in addition to, X-Amzn-</td>
</tr>
<tr>
<td>Header name</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MediaPackage-Endpoint-Id</td>
<td>MediaPackage-Endpoint-Id, to identify a given endpoint in the CDN logs. Endpoint names are unique only within a given channel and region. Using X-Amzn-Mediapackage-Endpoint-UniqueId is also helpful for support requests.</td>
<td></td>
</tr>
<tr>
<td>X-Amzn-RequestId</td>
<td>The unique identifier of the request.</td>
<td>Equivalent to X-Amzn-MediaPackage-Request-Id in MediaPackage V1. Using X-Amzn-RequestId is helpful for support requests.</td>
</tr>
</tbody>
</table>
Tagging AWS Elemental MediaPackage resources

A tag is a label that you assign to an AWS resource. Each tag consists of a key and a value, both of which you define. For example, the key might be "stage" and the value might be "test". You can use tags for a variety of purposes. One common use is to control access to AWS resources using tags. For information, see the Controlling access to AWS resources using tags topic in the IAM User Guide.

Another common use of tags is to categorize and track your MediaPackage costs. When you apply cost allocation tags to MediaPackage channels, endpoints, and packaging configurations, AWS generates a cost allocation report as a comma-separated value (CSV) file with your usage and costs aggregated by your tags. You can apply tags that represent business categories (such as cost centers, application names, or owners) to organize your costs across multiple services. For more information about using tags for cost allocation, see Using cost allocation tags in the AWS Billing User Guide.

Tag restrictions

The following restrictions apply to tagging AWS Elemental MediaPackage resources:

- Cost allocation tagging is only available for channel, endpoint, and packaging configuration resources. You can't use cost allocation tags for asset or packaging group 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.
- Can't be used for harvested live-to-VOD assets.

Managing tags

You can use the AWS Elemental MediaPackage API or the AWS CLI to add, edit, or delete the values for these properties.

For more information, see the actions related to tags in the following reference documentation:

- Tags resource-arm in the AWS Elemental MediaPackage live API reference.
- tag-resource in the AWS CLI MediaPackage reference.
Working with CDNs

You can use a content delivery network (CDN) such as Amazon CloudFront to serve the content that you store in AWS Elemental MediaPackage. A CDN is a globally distributed set of servers that caches content such as videos. When a user requests your content, the CDN routes the request to the edge location that provides the lowest latency. If your content is already cached in that edge location, the CDN delivers it immediately. If your content is not currently in that edge location, the CDN retrieves it from your origin (in this case, the MediaPackage endpoint) and distributes it to the user. The following illustration shows this process.

CDN configuration recommendations

To configure your CDN distributions for delivering Apple HLS and low-latency HLS (LL-HLS) streams with MediaPackage, we suggest using the following approach. Ensure that all your usual default parameters in the CDN distribution configuration are compatible with the delivery of HLS and LL-HLS streams with MediaPackage.
Honor MediaPackage 'cache-control: max-age' values

MediaPackage defines time-to-live (TTL) values for objects either statically or dynamically, depending on the object type. The specific TTLs are listed below, and it's strongly recommended that you honor these values and avoid overriding them at the CDN configuration level.

- Multivariant playlist (for both regular HLS and LL-HLS) - half the duration of the media segments
- Media playlists (regular HLS) - half the duration of the media segments
- Media playlists (LL-HLS) - 1 second
- TS media segments and init segments - 1209600 seconds (14 days)
- CMAF media segments and initialization segments - 1209600 seconds (14 days)

Include specific Query Strings in your CDN cache key

Some query strings are used to customize the manifests contents (aws.manifestfilter), to define specific startover content window (start and end), and to support LL-HLS playback requests logic (_HLS_msn, _HLS_part, and _HLS_skip). It's therefore recommended to include these six query strings in the CDN cache key. MediaPackage will ignore all other query strings, so don't include them in the CDN forward requests.

Response timeout

LL-HLS uses the Blocking Requests mechanism for both playlists and media parts, which is signaled through the EXT-X-PRELOAD-HINT tag. This mechanism puts the origin response on hold until the object is fully available. Consequently, the CDN should also wait for the origin response, and therefore, the response timeout value in your CDN distribution should be at least three times your parts duration.

Forwarded HTTP headers

You may consider including the Origin header in your CDN forward requests to MediaPackage as the only potentially necessary HTTP request header. In doing so, MediaPackage will respond with an access-control-allow-origin header, using the value passed as the Origin header value. If the Origin header is not included in the forward requests, MediaPackage will respond with an access-control-allow-origin: * header. Choose the approach that best fits your CORS requirements.

Forwarded cookies

MediaPackage doesn't consider any cookies that may be sent together with forward requests. Therefore, it's advisable to exclude cookies from CDN forward requests.
Quotas in AWS Elemental MediaPackage

The following sections provide information about the quotas in AWS Elemental MediaPackage.

Topics
- Live content quotas (p. 100)

Live content quotas

This section describes the quotas for live content in AWS Elemental MediaPackage. For information about requesting an increase to soft quotas, see AWS service quotas. You can't request an increase to hard quotas.

Live soft quotas

The following table describes quotas in AWS Elemental MediaPackage for live content that can be increased. For information about changing quotas, see AWS Service Quotas.

For some customers, your account quota might be below these published quotas. If you believe that you encountered a Resource limit exceeded error wrongfully, use the Service Quotas console to request quota increases.

<table>
<thead>
<tr>
<th>Resource or operation</th>
<th>Default quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Channel Groups</td>
<td>3 per account</td>
</tr>
<tr>
<td>Maximum Channels</td>
<td>10 per channel group</td>
</tr>
<tr>
<td>Maximum Endpoints per Channel</td>
<td>10 per channel</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Each origin endpoint represents the output package that you use. If one channel serves TS or TS encrypted, CMAF or CMAF encrypted content, then that channel has 4 endpoints and falls within the 10 endpoints quota. If you have 10 channels set up this same way, then you still haven't exceeded the quota because each channel uses only 4 endpoints.</td>
</tr>
</tbody>
</table>

Live hard quotas

The following table describes quotas in AWS Elemental MediaPackage for live content that can't be increased.

<table>
<thead>
<tr>
<th>Resource or operation</th>
<th>Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Live Manifest Length</td>
<td>15 minutes</td>
</tr>
<tr>
<td>Resource or operation</td>
<td>Quota</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Ingest streams per Channel</td>
<td>20 streams per channel</td>
</tr>
<tr>
<td>Maximum Content Age for Time-shifted Viewing</td>
<td>336 hours (14 days)</td>
</tr>
<tr>
<td>Maximum Time-shifted Manifest Length</td>
<td>24 hours for all supported output formats</td>
</tr>
<tr>
<td>Request Rates per Channel Input</td>
<td>200 requests per second</td>
</tr>
</tbody>
</table>
| Request Rates per Endpoint            | • Media segments output: 300 requests per second  
|                                       | • Manifests output: 500 requests per second |

**Note**

The per Endpoint origination request rate quotas are indicative only and based on typical traffic patterns when using a properly configured CDN. The request rate quotas are applicable for live events, linear channels, and time-shifted viewing. The request rate quotas may be lower under certain conditions like misconfigured CDNs or players generating abnormal levels of origin requests with unique HTTP headers values, or unique query strings values appended to the playback URLs.

<table>
<thead>
<tr>
<th>REST API Requests</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Steady state: 5 requests per second</td>
</tr>
<tr>
<td></td>
<td>• Bursting: 50 requests per second</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tracks per Ingest Stream</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The maximum number of tracks (audio, video, subtitle, etc.) per stream that you can ingest.</td>
</tr>
</tbody>
</table>
# AWS Elemental MediaPackage related information

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

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes and Workshops</td>
<td>Links to role-based and specialty courses and self-paced labs to help sharpen your AWS skills and gain practical experience.</td>
</tr>
<tr>
<td>AWS Developer Tools</td>
<td>Links to developer tools, SDKs, IDE tool kits, and command line tools for developing and managing AWS applications.</td>
</tr>
<tr>
<td>AWS Whitepapers</td>
<td>Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts.</td>
</tr>
<tr>
<td>AWS Support Center</td>
<td>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.</td>
</tr>
<tr>
<td>AWS Support</td>
<td>The primary web page for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.</td>
</tr>
<tr>
<td>Contact Us</td>
<td>A central contact point for inquiries concerning AWS billing, account, events, abuse, and other issues.</td>
</tr>
<tr>
<td>AWS Site Terms</td>
<td>Detailed information about our copyright and trademark; your account, license, and site access; and other topics.</td>
</tr>
</tbody>
</table>
## Document history for the MediaPackage User Guide

The following table describes the documentation releases for MediaPackage.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manifest filters (p. 29)</td>
<td>MediaPackage added support for manifest filters for endpoint egress requests.</td>
<td>October 30, 2023</td>
</tr>
<tr>
<td>Response headers (p. 95)</td>
<td>Added section that explains MediaPackage response headers.</td>
<td>August 4, 2023</td>
</tr>
<tr>
<td>AWS managed policy updates (p. 75)</td>
<td>Added new AWS managed policies: AWSElementalMediaPackageV2FullAccess, AWSElementalMediaPackageV2ReadOnly.</td>
<td>July 25, 2023</td>
</tr>
<tr>
<td>time_delay query parameters added to time-shifted viewing reference (p. 50)</td>
<td>Added information on using time_delay query parameters.</td>
<td>July 23, 2023</td>
</tr>
<tr>
<td>Added data plane APIs (p. 53)</td>
<td>Added PutObject, and data plane APIs.</td>
<td>June 14, 2023</td>
</tr>
<tr>
<td>Initial release (p. 103)</td>
<td>Initial release of the AWS Elemental MediaPackage User Guide</td>
<td>May 5, 2023</td>
</tr>
</tbody>
</table>
AWS Glossary

For the latest AWS terminology, see the AWS glossary in the AWS Glossary Reference.