Developer Guide

Amazon Kinesis Video Streams

Amazon Kinesis Video Streams: Developer Guide

Table of Contents

What is Kinesis Video Streams?

You can use Amazon Kinesis Video Streams, a fully managed AWS service, to stream live video from devices to the AWS Cloud, or build applications for real-time video processing or batch-oriented video analytics.

Kinesis Video Streams isn't only storage for video data. You can use it to watch your video streams in real time as they are received in the cloud. You can either monitor your live streams in the AWS Management Console, or develop your own monitoring application that uses the Kinesis Video Streams API library to display live video.

You can use Kinesis Video Streams to capture massive amounts of live video data from millions of sources, including smartphones, security cameras, webcams, cameras embedded in cars, drones, and other sources. You can also send non-video, time-serialized data such as audio data, thermal imagery, depth data, and RADAR data. As live video streams from these sources into a Kinesis video stream, you can build applications to access the data, frame-by-frame, in real time for low-latency processing. Kinesis Video Streams is source-agnostic. You can stream video from a computer's webcam using the GStreamer Plugin - kvssink library, or from a camera on your network using real-time streaming protocol (RTSP).

You can also configure your Kinesis video stream to durably store media data for the specified retention period. Kinesis Video Streams automatically stores this data and encrypts it at rest. Additionally, Kinesis Video Streams time-indexes stored data based on both the producer timestamps and ingestion timestamps. You can build applications that periodically batch-process the video data, or you can create applications that require one-time access to historical data for different use cases.

Your custom applications, real-time or batch-oriented, can run on Amazon EC2 instances. These applications might process data using open source, deep-learning algorithms, or use third-party applications that integrate with Kinesis Video Streams.

Benefits of using Kinesis Video Streams include the following:

- **Connect and stream from millions of devices –** You can use Kinesis Video Streams to connect and stream video, audio, and other data from millions of devices ranging from consumer smartphones, drones, and dash cams. You can use the Kinesis Video Streams producer libraries to configure your devices and reliably stream in real time, or as after-the-fact media uploads.

- **Durably store, encrypt, and index data –** You can configure your Kinesis video stream to durably store media data for custom retention periods. Kinesis Video Streams also generates an index over the stored data based on producer-generated or service-side timestamps. Your applications can retrieve specified data in a stream using the time-index.
- **Focus on managing applications instead of infrastructure –** Kinesis Video Streams is serverless, so there's no infrastructure to set up or manage. You don't need to worry about the deployment, configuration, or elastic scaling of the underlying infrastructure because your data streams and number of consuming applications grow and shrink. Kinesis Video Streams automatically does all the administration and maintenance required to manage streams, so you can focus on the applications, not the infrastructure.
- **Build real-time and batch applications on data streams –** You can use Kinesis Video Streams to build custom real-time applications that operate on live data streams, and create batch or one-time applications that operate on durably persisted data without strict latency requirements. You can build, deploy, and manage custom applications: open source (Apache MXNet, OpenCV), homegrown, or third-party solutions using the AWS Marketplace to process and analyze your streams. You can use Kinesis Video Streams `Get` APIs to build multiple concurrent applications processing data in a real-time or batch-oriented basis.
- **Stream data more securely –** Kinesis Video Streams encrypts all data as it flows through the service and when it persists the data. Kinesis Video Streams enforces Transport Layer Security (TLS)-based encryption on data streaming from devices, and encrypts all data at rest using AWS Key Management Service (AWS KMS). Additionally, you can manage access to your data using AWS Identity and Access Management (IAM).
- **Pay as you go –** For more information, see AWS Pricing Calculator.

Region availability

Amazon Kinesis Video Streams is available in the following regions:

Region Name	AWS Region Code
US East (Ohio)	us-east-2
US East (N. Virginia)	us-east-1
US West (Oregon)	us-west-2

Region Name	AWS Region Code
Africa (Cape Town)	af-south-1
Asia Pacific (Hong Kong)	ap-east-1
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
China (Beijing)	cn-north-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
South America (Sao Paulo)	sa-east-1

Are you a first-time user of Kinesis Video Streams?

If you're a first-time user of Kinesis Video Streams, we recommend that you read the following sections in order:

1. **Kinesis Video Streams: How it works** – To learn about Kinesis Video Streams concepts.
2. **Getting started with Amazon Kinesis Video Streams** – To set up your account and test Kinesis Video Streams.
3. **Kinesis Video Streams Producer Libraries** – To learn about creating a Kinesis Video Streams producer application.

4. **Kinesis video stream Parser Library** – To learn about processing incoming data frames in a Kinesis Video Streams consumer application.
5. **Amazon Kinesis Video Streams examples** – To see more examples of what you can do with Kinesis Video Streams.

Kinesis Video Streams system requirements

The following sections contain hardware, software, and storage requirements for Amazon Kinesis Video Streams.

Topics

Camera requirements

Cameras that are used for running the Kinesis Video Streams Producer SDK and samples have the following memory requirements:

- The SDK content view requires 16 MB of memory.
- The sample application default configuration is 128 MiB of memory. This value is appropriate for producers that have good network connectivity and no requirements for additional buffering. If the network connectivity is poor and more buffering is required, you can calculate the memory requirement per second of buffering by multiplying the frame rate per second by the frame memory size. For more information about allocating memory, see StorageInfo.

We recommend using USB or RTSP (Real Time Streaming Protocol) cameras that encode data using H.264 because this removes the encoding workload from the CPU.

Currently, the demo application doesn't support the User Datagram Protocol (UDP) for RTSP streaming. This capability will be added in the future.

The Producer SDK supports the following types of cameras:

- Web cameras.
- USB cameras.
- Cameras with H.264 encoding (preferred).
- Cameras without H.264 encoding.

- Raspberry Pi camera module. This is preferred for Raspberry Pi devices because it connects to the GPU for video data transfer, so there is no overhead for CPU processing.
- RTSP (network) cameras. These cameras are preferred because the video streams are already encoded with H.264.

Tested operating systems

We have tested web cameras and RTSP cameras with the following devices and operating systems:

- Mac mini
  - High Sierra
- MacBook Pro laptops
  - Sierra (10.12)
  - El Capitan (10.11)
- HP laptops running Ubuntu 16.04
- Ubuntu 17.10 (Docker container)
- Raspberry Pi 3

SDK storage requirements

Installing the [Kinesis Video Streams Producer Libraries] has a minimum storage requirement of 170 MB and a recommended storage requirement of 512 MB.

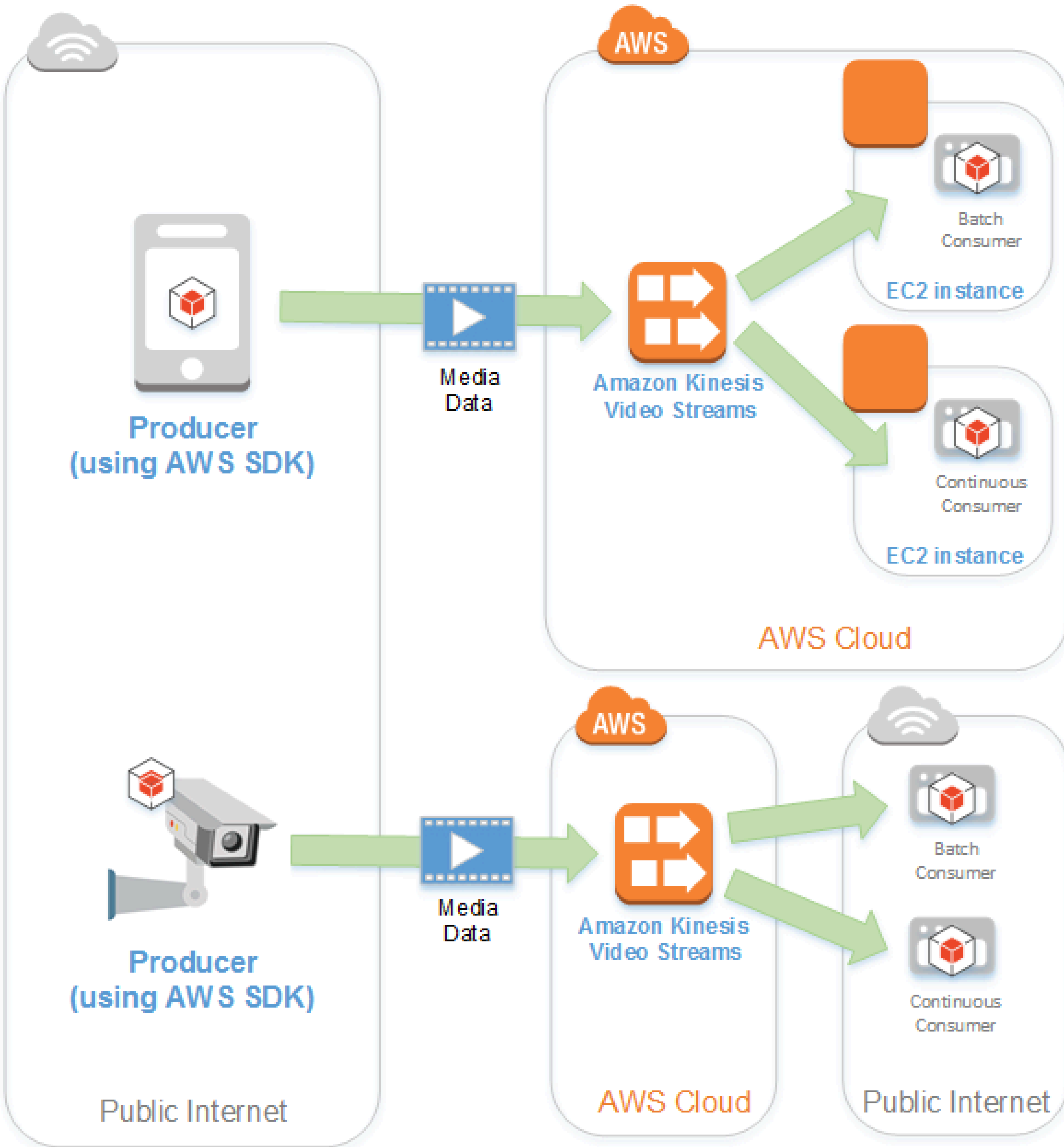
Kinesis Video Streams: How it works

Topics

- Kinesis Video Streams API and producer libraries support
- Kinesis Video Streams playback
- Using streaming metadata with Kinesis Video Streams
- Kinesis Video Streams data model

You can use Amazon Kinesis Video Streams, a fully managed AWS service, to stream live video from devices to the AWS Cloud and durably store it. You can then build your own applications for real-time video processing or perform batch-oriented video analytics.

The following diagram provides an overview of how Kinesis Video Streams works.



The diagram demonstrates the interaction among the following components:

- **Producer** – Any source that puts data into a Kinesis video stream. A producer can be any video-generating device, such as a security camera, a body-worn camera, a smart phone camera, or a dashboard camera. A producer can also send non-video data, such as audio feeds, images, or RADAR data.

  A single producer can generate one or more video streams. For example, a video camera can push video data to one Kinesis video stream and audio data to another.

  - **Kinesis Video Streams Producer libraries –** A set of software and libraries that you can install and configure on your devices. You can use these libraries to securely connect and reliably stream video in different ways, including in real time, after buffering it for a few seconds, or as after-the-fact media uploads.

- **Kinesis video stream –** A resource that you can use to transport live video data, optionally store it, and make the data available for consumption both in real time and on a batch or one-time basis. In a typical configuration, a Kinesis video stream has only one producer publishing data into it.

  The stream can carry audio, video, and similar time-encoded data streams, such as depth sensing feeds, RADAR feeds, and more. You create a Kinesis video stream using the AWS Management Console or programmatically using the AWS SDKs.

  Multiple independent applications can consume a Kinesis video stream in parallel.

- **Consumer –** Gets data, such as fragments and frames, from a Kinesis video stream to view, process, or analyze it. Generally these consumers are called Kinesis Video Streams applications. You can write applications that consume and process data in Kinesis Video Streams in real time, or after the data is stored and time-indexed when low latency processing isn't required. You can create these consumer applications to run on Amazon EC2 instances.

  - Kinesis video stream Parser Library – Enables Kinesis Video Streams applications to reliably get media from Kinesis video stream in a low-latency manner. Additionally, it parses the frame boundaries in the media so that applications can focus on processing and analyzing the frames themselves.

Kinesis Video Streams API and producer libraries support

Kinesis Video Streams provides APIs for you to create and manage streams and read or write media data to and from a stream. The Kinesis Video Streams console, in addition to administration functionality, also supports live and video-on-demand playback. Kinesis Video Streams also provides a set of producer libraries that you can use in your application code to extract data from your media sources and upload to your Kinesis video stream.

Topics

Kinesis Video Streams API

Kinesis Video Streams provides APIs for creating and managing Kinesis Video Streams. It also provides APIs for reading and writing media data to a stream, as follows:

- **Producer API –** Kinesis Video Streams provides a `PutMedia` API to write media data to a Kinesis video stream. In a `PutMedia` request, the producer sends a stream of media fragments. A *fragment* is a self-contained sequence of frames. The frames belonging to a fragment should have no dependency on any frames from other fragments. For more information, see PutMedia.

  As fragments arrive, Kinesis Video Streams assigns a unique fragment number, in increasing order. It also stores producer-side and server-side timestamps for each fragment, as Kinesis Video Streams-specific metadata.
- **Consumer APIs –** Consumers can use the following APIs to get data from a stream:
  - `GetMedia` - When using this API, consumers must identify the starting fragment. The API then returns fragments in the order in which they were added to the stream (in increasing order by fragment number). The media data in the fragments is packed into a structured format such as Matroska (MKV). For more information, see GetMedia.

    > **Note**
    >
    > `GetMedia` knows where the fragments are (archived in the data store or available in real time). For example, if `GetMedia` determines that the starting fragment is archived, it starts returning fragments from the data store. When it must return newer fragments that aren't archived yet, `GetMedia` switches to reading fragments from an in-memory stream buffer.

    This is an example of a continuous consumer, which processes fragments in the order that they are ingested by the stream.

    `GetMedia` enables video-processing applications to fail or fall behind, and then catch up with no additional effort. Using `GetMedia`, applications can process data that's archived in the data store, and as the application catches up, `GetMedia` continues to feed media data in real time as it arrives.
  - `GetMediaFromFragmentList` (and `ListFragments`) - Batch processing applications are considered offline consumers. Offline consumers might choose to explicitly fetch

particular media fragments or ranges of video by combining the `ListFragments` and `GetMediaFromFragmentList` APIs. `ListFragments` and `GetMediaFromFragmentList` enable an application to identify segments of video for a particular time range or fragment range, and then fetch those fragments either sequentially or in parallel for processing. This approach is suitable for `MapReduce` application suites, which must quickly process large amounts of data in parallel.

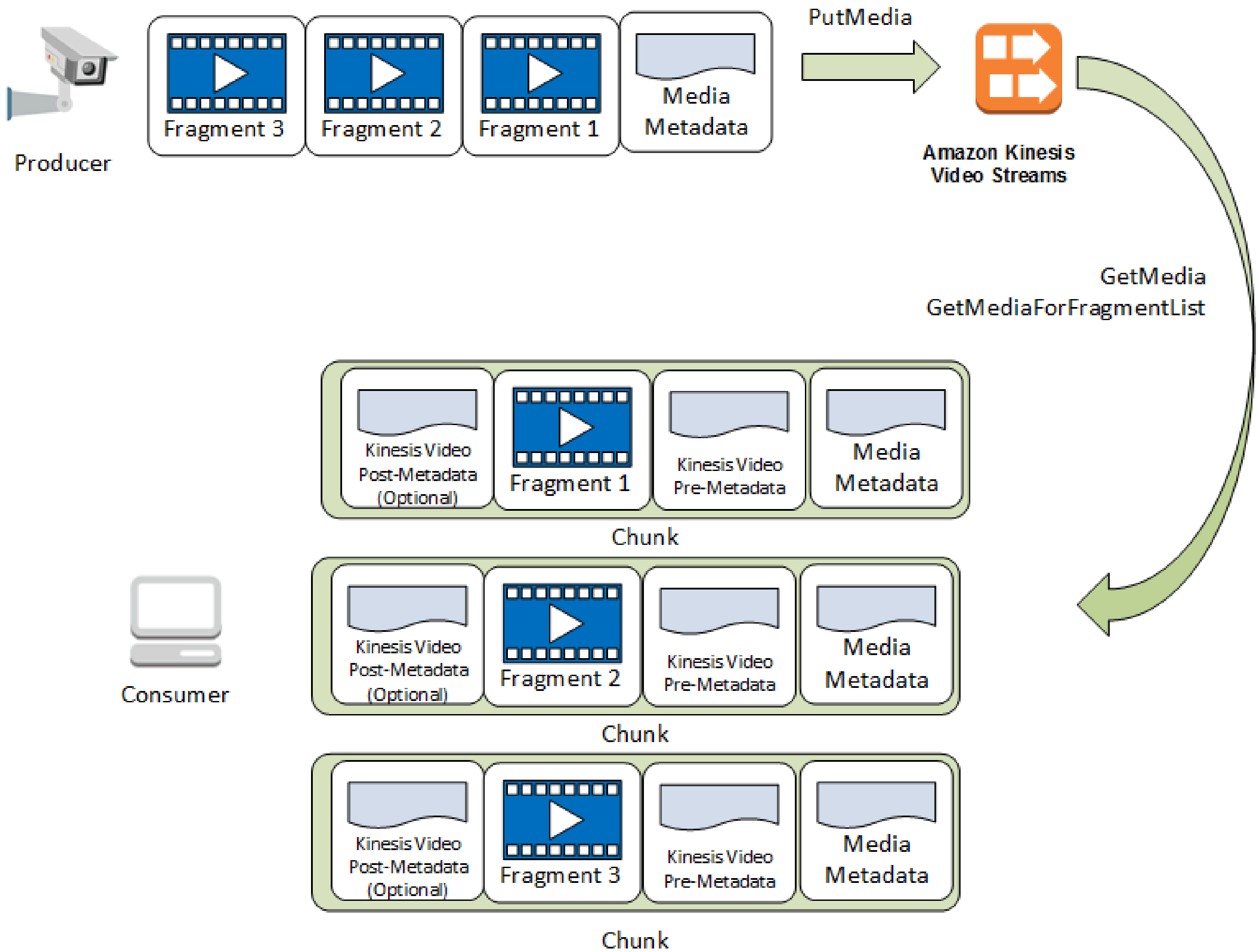
For example, suppose that a consumer wants to process one day's worth of video fragments. The consumer would do the following:

1. Get a list of fragments by calling the `ListFragments` API and specifying a time range to select the desired collection of fragments.

   The API returns metadata from all the fragments in the specified time range. The metadata provides information such as fragment number, producer-side and server-side timestamps, and so on.

2. Take the fragment metadata list and retrieve fragments, in any order. For example, to process all the fragments for the day, the consumer might choose to split the list into sublists and have workers (for example, multiple Amazon EC2 instances) fetch the fragments in parallel using the `GetMediaFromFragmentList`, and process them in parallel.

The following diagram shows the data flow for fragments and chunks during these API calls.



When a producer sends a `PutMedia` request, it sends media metadata in the payload, and then sends a sequence of media data fragments. Upon receiving the data, Kinesis Video Streams stores incoming media data as Kinesis Video Streams chunks. Each chunk consists of the following:

- A copy of the media metadata
- A fragment
- Kinesis Video Streams-specific metadata; for example, the fragment number and server-side and producer-side timestamps

When a consumer requests media metadata, Kinesis Video Streams returns a stream of chunks, starting with the fragment number that you specify in the request.

If you enable data persistence for the stream, after receiving a fragment on the stream, Kinesis Video Streams also saves a copy of the fragment to the data store.

Endpoint discovery pattern

Control Plane REST APIs

To access the Kinesis Video Streams Control Plane REST APIs, use the Kinesis Video Streams service endpoints.

Data Plane REST APIs

Kinesis Video Streams is built using a cellular architecture to ensure better scaling and traffic isolation properties. Because each stream is mapped to a specific cell in a region, your application must use the correct cell-specific endpoints that your stream has been mapped to. When accessing the Data Plane REST APIs, you will need to manage and map the correct endpoints yourself. This process, the endpoint discovery pattern, is described below:

1. The endpoint discovery pattern starts with a call to one of the `GetEndpoints` actions. These actions belong to the Control Plane.
    1. If you are retrieving the endpoints for the the section called "Amazon Kinesis Video Streams Media" or the section called "Amazon Kinesis Video Streams Archived Media" services, use the section called "GetDataEndpoint".
    2. If you are retrieving the endpoints for the section called "Amazon Kinesis Video Signaling Channels", the section called "Amazon Kinesis Video WebRTC Storage", or Kinesis Video Signaling, use the section called "GetSignalingChannelEndpoint".
2. Cache and reuse the endpoint.
3. If the cached endpoint no longer works, make a new call to `GetEndpoints` to refresh the endpoint.

Producer libraries

After you create a Kinesis video stream, you can start sending data to the stream. In your application code, you can use these libraries to extract data from your media sources and upload to your Kinesis video stream. For more information about the available producer libraries, see Kinesis Video Streams Producer Libraries.

Kinesis Video Streams playback

You can view a Kinesis video stream using the following methods:

- **GetMedia** – You can use the `GetMedia` API to build your own applications to process Kinesis Video Streams. `GetMedia` is a real-time API with low latency. To create a player that uses `GetMedia`, you must build it yourself. For information about how to develop an application that displays a Kinesis video stream using `GetMedia`, see Stream Parser Library.
- **HLS** – HTTP Live Streaming (HLS) is an industry standard HTTP-based media streaming communications protocol. You can use HLS to view a Kinesis video stream, either for live playback or to view archived video.

  You can use HLS for live playback. Latency is typically between 3–5 seconds, but it can be between 1–10 seconds, depending on the use case, player, and network conditions. You can use a third-party player (such as Video.js or Google Shaka Player) to display the video stream by providing the HLS streaming session URL, either programmatically or manually. You can also play back video by entering the HLS streaming session URL in the **Location** bar of the Apple Safari or Microsoft Edge browsers.
- **MPEG-DASH** – Dynamic Adaptive Streaming over HTTP (DASH), also known as MPEG-DASH, is an adaptive bitrate streaming protocol that enables high quality streaming of media content over the internet delivered from conventional HTTP web servers.

  You can use MPEG-DASH for live playback. Latency is typically between 3–5 seconds, but it can be between 1–10 seconds, depending on the use case, player, and network conditions. You can use a third-party player (such as dash.js or Google Shaka Player) to display the video stream by providing the MPEG-DASH streaming session URL, either programmatically or manually.
- **GetClip** – You can use the `GetClip` API to download a clip (in an MP4 file) containing the archived, on-demand media from the specified video stream over the specified time range. For more information, see the GetClip API Reference.

Topics

Video playback track requirements

Amazon Kinesis Video Streams supports media encoded in multiple formats. If your Kinesis video stream uses a format not supported by one of the four APIs listed below, use `GetMedia` or `GetMediaForFragmentList`, as they don't have track-type limitations.

Topics

GetClip requirements

For more information about this API, see `GetClip`.

Track 1 description	Track 1 codec ID	Track 2 description	Track 2 codec ID
H.264 video	V_MPEG/ISO/AVC	N/A	N/A
H.264 video	V_MPEG/ISO/AVC	AAC audio	A_AAC
H.264 video	V_MPEG/ISO/AVC	G.711 audio (A-Law only)	A_MS/ACM
H.265 video	V_MPEGH/ISO/HEVC	N/A	N/A
H.265 video	V_MPEGH/ISO/HEVC	AAC audio	A_AAC

> **Important**
>
> The codec private data (CPD) contained in each fragment contains codec-specific initialization information, such as frame rate, resolution, and encoding profile, which are necessary to properly decode the fragment. CPD changes aren't supported between the target fragments of the resulting clip. The CPD must remain consistent through the queried media, otherwise an error will be returned.

> **Important**
>
> Track changes aren't supported. Tracks must remain consistent throughout the queried media. An error is returned if the fragments in the stream change from having only video to having both audio and video, or if an AAC audio track is changed to an A-Law audio track.

GetDASHStreamingSessionURL requirements

For more information about this API, see `GetDASHStreamingSessionURL`.

Track 1 description	Track 1 codec ID	Track 2 description	Track 2 codec ID
H.264 video	V_MPEG/ISO/AVC	N/A	N/A
H.264 video	V_MPEG/ISO/AVC	AAC audio	A_AAC
H.264 video	V_MPEG/ISO/AVC	G.711 audio (A-Law only)	A_MS/ACM
H.264 video	V_MPEG/ISO/AVC	G.711 audio (U-Law only)	A_MS/ACM
AAC audio	A_AAC	N/A	N/A
H.265 video	V_MPEGH/ISO/HEVC	N/A	N/A
H.265 video	V_MPEGH/ISO/HEVC	AAC audio	A_AAC

> **Important**
>
> The codec private data (CPD) contained in each fragment contains codec-specific initialization information, such as frame rate, resolution, and encoding profile, which are necessary to properly decode the fragment. CPD changes aren't supported during a streaming session. The CPD must remain consistent through the queried media.

> **Important**
>
> Track changes aren't supported. Tracks must remain consistent throughout the queried media. Streaming will fail if the fragments in the stream change from having only video to having both audio and video, or if an AAC audio track is changed to an A-Law audio track.

GetHLSStreamingSessionURL requirements

For more information about this API, see `GetHLSStreamingSessionURL`.

HLS Mp4

Track 1 description	Track 1 codec ID	Track 2 description	Track 2 codec ID
H.264 video	V_MPEG/ISO/AVC	N/A	N/A
H.264 video	V_MPEG/ISO/AVC	AAC audio	A_AAC
AAC audio	A_AAC	N/A	N/A
H.265 video	V_MPEGH/ISO/HEVC	N/A	N/A
H.265 video	V_MPEGH/ISO/HEVC	AAC audio	A_AAC

HLS TS

Track 1 description	Track 1 codec ID	Track 2 description	Track 2 codec ID
H.264 video	V_MPEG/ISO/AVC	N/A	N/A
H.264 video	V_MPEG/ISO/AVC	AAC audio	A_AAC
AAC audio	A_AAC	N/A	N/A

> **Note**
>
> The codec private data (CPD) contained in each fragment contains codec-specific initialization information, such as frame rate, resolution, and encoding profile, which

are necessary to properly decode the fragment. For both TS and MP4, CPD changes are supported during a streaming session. Therefore, the fragments in a session can have a different information in the CPD without interrupting playback. For each streaming session, only 500 CPD changes are allowed.

Important

Track changes aren't supported. Tracks must remain consistent throughout the queried media. Streaming will fail if the fragments in the stream change from having only video to having both audio and video, or if an AAC audio track is changed to an A-Law audio track.

GetImages requirements

For more information about this API, see `GetImages`.

Note

The `GetImages` media should contain a video track in track 1.

Video playback with HLS

HTTP Live Streaming (HLS) is an industry standard HTTP-based media streaming communications protocol. You can use HLS to view a Kinesis video stream, either for live playback or to view archived video.

You can use HLS for live playback. Latency is typically between 3–5 seconds, but it can be between 1–10 seconds, depending on the use case, player, and network conditions. You can use a third-party player (such as Video.js or Google Shaka Player) to display the video stream by providing the HLS streaming session URL, either programmatically or manually. You can also play back video by entering the HLS streaming session URL in the Location bar of the Apple Safari or Microsoft Edge browsers.

To view a Kinesis video stream using HLS, first create a streaming session using GetHLSStreamingSessionURL. This action returns a URL (containing a session token) for accessing the HLS session. You can then use the URL in a media player or a standalone application to display the stream.

> **Important**
>
> Not all media sent to Kinesis Video Streams can be played back through HLS. See the section called “GetHLSStreamingSessionURL” for specific uploading requirements.

Topics

- Use the AWS CLI to retrieve an HLS streaming session URL
- Example: Use HLS in HTML and JavaScript
- Troubleshooting HLS issues

Use the AWS CLI to retrieve an HLS streaming session URL

Follow the procedure below to use the AWS CLI to generate an HLS streaming session URL for a Kinesis video stream.

For installation instructions, see the AWS Command Line Interface User Guide. After installation, configure the AWS CLI with credentials and region.

Alternatively, open the AWS CloudShell terminal, which has the AWS CLI installed and configured. See the AWS CloudShell User Guide for more information.

Retrieve the HLS URL endpoint for your Kinesis video stream.

1. Type the following into the terminal:

   ```
   aws kinesisvideo get-data-endpoint \
     --api-name GET_HLS_STREAMING_SESSION_URL \
     --stream-name YourStreamName
   ```

   You'll receive a response that looks like this:

   ```
   {
       "DataEndpoint": "https://b-1234abcd.kinesisvideo.aws-region.amazonaws.com"
   }
   ```

2. Make the HLS streaming session URL request to that returned endpoint.

   Live

   For live playback, the HLS media playlist is continually updated with the latest media as it becomes available. When you play this type of session in a media player, the user interface typically displays a "**live**" notification, with no scrubber control for choosing the position in the playback window to display.

   Make sure that you are uploading media to this stream when you run this command.

```
aws kinesis-video-archived-media get-hls-streaming-session-url \
  --endpoint-url https://b-1234abcd.kinesisvideo.aws-region.amazonaws.com \
  --stream-name YourStreamName \
  --playback-mode LIVE
```

   Live replay

   For live replay, playback starts from a specified start time. The HLS media playlist is also continually updated with the latest media as it becomes available. The session will continue to include newly ingested media until the session expires, or until the specified end time, whichever comes first. This mode is useful to be able to start playback from when an event is detected and continue live streaming media that has not yet been ingested as of the time of the session creation.

   Determine a start timestamp.

   For this example, we use the **Unix Epoch time in seconds** format. Refer to the Timestamps section in the AWS Command Line Interface User Guide for more information on timestamp formatting.

   See UnixTime.org for a conversion tool.

   - **1708471800** is equal to **February 20, 2024 3:30:00 PM GMT-08:00**

   In this example, we don't specify an end timestamp, meaning that the session will continue to include newly ingested media until the session expires.

   Invoke the `GetHLSStreamingSessionURL` API with `LIVE_REPLAY` playback mode and an HLS Fragment Selector specified.

```
aws kinesis-video-archived-media get-hls-streaming-session-url \
  --endpoint-url https://b-1234abcd.kinesisvideo.aws-region.amazonaws.com \
  --stream-name YourStreamName \
  --playback-mode LIVE_REPLAY \
  --hls-fragment-selector \

 "FragmentSelectorType=SERVER_TIMESTAMP,TimestampRange={StartTimestamp=1708471800}"
```

On-demand

For on demand playback, the HLS media playlist contains the media specified by the HLS fragment selector. When this type of session is played in a media player, the user interface typically displays a scrubber control for choosing the position in the playback window to display.

To create a URL for a certain section of stream, first determine start and end timestamps.

For this example, we use the **Unix Epoch time in seconds** format. Refer to the Timestamps section in the AWS Command Line Interface User Guide for more information on timestamp formatting.

See UnixTime.org for a conversion tool.

- **1708471800** is equal to **February 20, 2024 3:30:00 PM GMT-08:00**
- **1708471860** is equal to **February 20, 2024 3:31:00 PM GMT-08:00**

Invoke the `GetHLSStreamingSessionURL` API with `ON_DEMAND` playback mode and an HLS Fragment Selector specified.

```
aws kinesis-video-archived-media get-hls-streaming-session-url \
  --endpoint-url https://b-1234abcd.kinesisvideo.aws-region.amazonaws.com \
  --stream-name YourStreamName \
  --playback-mode ON_DEMAND \
  --hls-fragment-selector \

 "FragmentSelectorType=SERVER_TIMESTAMP,TimestampRange={StartTimestamp=1708471800,EndTim
```

> **Note**
>
> The timestamps must be within 24-hours of each other, as mentioned in the the section called "HLSTimestampRange" documentation.

You'll receive a response that looks like this:

```
{
    "HLSStreamingSessionURL": "https://b-1234abcd.kinesisvideo.aws-region.amazonaws.com/hls/v1/getHLSMasterPlaylist.m3u8?SessionToken=CiAz...DkRE6M~"
}
```

> **Important**
>
> Don't share or store this token where an unauthorized entity could access it. The token provides access to the content of the stream. Safeguard the token with the same measures that you would use with your AWS credentials.

You can use this URL and any HLS player to view the HLS stream.

For example, use VLC media player.

You can also play the HLS Stream by entering the HLS streaming session URL in the Location bar of the Apple Safari or Microsoft Edge browsers.

Example: Use HLS in HTML and JavaScript

The following example shows how to use the AWS SDK for JavaScript v2 to retrieve an HLS streaming session for a Kinesis video stream and play it back in a web page. The example shows how to play back video in the following players:

- Video.js
- Google Shaka Player
- hls.js

View the complete example code and hosted web page in GitHub.

Code walk through topics:

- Import the AWS SDK for JavaScript for browsers
- Set up the Kinesis Video Streams client
- Retrieve the endpoint for HLS playback
- Set up the Kinesis Video Streams archived media client
- Retrieve the HLS streaming session URL
- Display the HLS stream on the web page

Import the AWS SDK for JavaScript for browsers

In the web page, include the following script tag to import the AWS SDK for JavaScript v2 into the project.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/aws-sdk/2.490.0/aws-sdk.min.js"></
script>
```

For more information, refer to the AWS SDK for JavaScript documentation.

Set up the Kinesis Video Streams client

To access streaming video with HLS, first create and configure the Kinesis Video Streams client. See Setting Credentials in a Web Browser for other authentication methods.

```
const clientConfig = {
    accessKeyId: 'YourAccessKey',
    secretAccessKey: 'YourSecretKey',
    region: 'us-west-2'
};
const kinesisVideoClient = new AWS.KinesisVideo(clientConfig);
```

The application retrieves the necessary values from input boxes on the HTML page.

Retrieve the endpoint for HLS playback

Use the Kinesis Video Streams client to invoke the the section called “GetDataEndpoint” API to retrieve the endpoint.

```
const getDataEndpointOptions = {
    StreamName: 'YourStreamName',
    APIName: 'GET_HLS_STREAMING_SESSION_URL'
};
const getDataEndpointResponse = await kinesisVideoClient
    .getDataEndpoint(getDataEndpointOptions)
    .promise();
const hlsDataEndpoint = getDataEndpointResponse.DataEndpoint;
```

This code stores the endpoint in the `hlsDataEndpoint` variable.

Set up the Kinesis Video Streams archived media client

In the client configuration for the Kinesis Video Streams archived media client, specify the endpoint that you obtained in the previous step.

```
const archivedMediaClientConfig = {
    accessKeyId: 'YourAccessKey',
    secretAccessKey: 'YourSecretKey',
    region: 'us-west-2',
    endpoint: hlsDataEndpoint
};
const kinesisVideoArchivedMediaClient = new
 AWS.KinesisVideoArchivedMedia(archivedMediaClientConfig);
```

Retrieve the HLS streaming session URL

Use the Kinesis Video Streams archived media client to invoke the the section called "GetHLSStreamingSessionURL" API to retrieve the HLS playback URL.

```
const getHLSStreamingSessionURLOptions = {
    StreamName: 'YourStreamName',
    PlaybackMode: 'LIVE'
};
const getHLSStreamingSessionURLResponse = await kinesisVideoArchivedMediaClient
    .getHLSStreamingSessionURL(getHLSStreamingSessionURLOptions)
    .promise();
const hlsUrl = getHLSStreamingSessionURLResponse.HLSStreamingSessionURL;
```

Display the HLS stream on the web page

When you have the HLS streaming session URL, provide it to the video player. The method for providing the URL to the video player is specific to the player used.

Video.js

Do the following to import Video.js and its CSS classes into our browser script:

```
<link rel="stylesheet" href="https://vjs.zencdn.net/6.6.3/video-js.css">
<script src="https://vjs.zencdn.net/6.6.3/video.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/videojs-contrib-hls/5.14.1/
videojs-contrib-hls.js"></script>
```

Create a `video` HTML element to display the video:

```
<video id="videojs" class="player video-js vjs-default-skin" controls autoplay></
video>
```

Set the HLS URL as the HTML video element source:

```
const playerElement = document.getElementById('videojs');
const player = videojs(playerElement);
player.src({
    src: hlsUrl,
    type: 'application/x-mpegURL'
});
player.play();
```

Shaka

Do the following to import the Google Shaka player into our browser script:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/shaka-player/2.4.1/shaka-
player.compiled.js"></script>
```

Create a `video` HTML element to display the video:

```
<video id="shaka" class="player" controls autoplay></video>
```

Create a Shaka player specifying the video element and call the load method:

```
const playerElement = document.getElementById('shaka');
const player = new shaka.Player(playerElement);
player.load(hlsUrl);
```

hls.js

Do the following to import hls.js into our browser script:

```
<script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>
```

Create a `video` HTML element to display the video:

```
<video id="hlsjs" class="player" controls autoplay></video>
```

Create an hls.js player, give it the HLS URL, and tell it to play:

```
const playerElement = document.getElementById('hlsjs');
const player = new Hls();
player.loadSource(hlsUrl);
player.attachMedia(playerElement);
player.on(Hls.Events.MANIFEST_PARSED, function() {
    video.play();
});
```

Troubleshooting HLS issues

This section describes issues that you might encounter when using HTTP Live Streaming (HLS) with Kinesis Video Streams.

Issues

- Retrieving HLS streaming session URL succeeds, but playback fails in video player
- Latency too high between producer and player

Retrieving HLS streaming session URL succeeds, but playback fails in video player

This situation occurs when you can successfully retrieve an HLS streaming session URL using `GetHLSStreamingSessionURL`, but the video fails to play back when the URL is provided to a video player.

To troubleshoot this situation, try the following:

- Determine whether the video stream plays back in the Kinesis Video Streams console. Consider any errors that the console shows.
- If the fragment duration is less than one second, increase it to one second. If the fragment duration is too short, the service might throttle the player because it's making requests for video fragments too frequently.
- Verify that each HLS streaming session URL is being used by only one player. If more than one player is using a single HLS streaming session URL, the service might receive too many requests and throttle them.
- Verify that your player supports all of the options that you're specifying for the HLS streaming session. Try different combinations of values for the following parameters:
  - `ContainerFormat`
  - `PlaybackMode`
  - `FragmentSelectorType`
  - `DiscontinuityMode`
  - `MaxMediaPlaylistFragmentResults`

  Some media players (like HTML5 and mobile players) typically only support HLS with the fMP4 container format. Other media players (like Flash and custom players) might only support HLS with the MPEG TS container format. We recommend experimenting with the `ContainerFormat` parameter to start troubleshooting.
- Verify that each fragment has a consistent number of tracks. Verify that fragments in the stream are not changing between having both an audio and video track and only a video track. Also verify that the encoder settings (resolution and frame rate) are not changing between fragments in each track.

Latency too high between producer and player

This situation occurs when the latency is too high from when the video is captured to when it is played in the video player.

Video is played back through HLS on a per-fragment basis. Therefore, latency can't be less than fragment duration. Latency also includes the time needed for buffering and transferring data. If your solution requires latency of less than one second, consider using the `GetMedia` API instead.

You can adjust the following parameters to reduce the overall latency, but adjusting these parameters might also reduce the video quality or increase the rebuffering rate.

- **Fragment duration** – The fragment duration is the amount of video between divisions in the stream as controlled by the frequency of keyframes generated by the video encoder. The recommended value is one second. Having a shorter fragment duration means that less time is spent waiting for the fragment to complete before transmitting the video data to the service. Shorter fragments are also faster for the service to process. However, if the fragment duration is too short, the probability increases that the player will run out of content and have to stop and buffer content. If the fragment duration is less than 500 milliseconds, the producer might create too many requests, causing the service to throttle them.
- **Bitrate** – A video stream with a lower bitrate takes less time to read, write, and transmit. However, a video stream with a lower bitrate usually has a lower video quality.
- **Fragment count in media playlists** – A latency-sensitive player should only load the newest fragments in a media playlist. Most players start at the earliest fragment instead. By reducing the number of fragments in the playlist, you reduce the time separation between the previous and new fragments. With a smaller playlist size, it's possible for a fragment to be skipped during playback if there's a delay in adding new fragments to the playlist, or if there's a delay in the player getting an updated playlist. We recommend using 3–5 fragments, and to use a player that's configured to load only the newest fragments from a playlist.
- **Player buffer size** – Most video players have a configurable minimum buffer duration, usually with a 10-second default. For the lowest latency, you can set this value to 0 seconds. However, doing so means that the player rebuffers if there's any delay producing fragments because the player will have no buffer for absorbing the delay.
- **Player "catch up"** – Video players typically don't automatically catch playback up to the front of the video buffer if the buffer fills up, like when a delayed fragment causes a backlog of fragments to play. A custom player can avoid this by either dropping frames, or increasing the playback speed (for example, to 1.1x) to catch up to the front of the buffer. This causes playback to be choppy or increase in speed as the player catches up, and rebuffering might be more frequent as the buffer size is kept short.

Video Playback with MPEG-DASH

To view a Kinesis video stream using MPEG-DASH, you first create a streaming session using GetDASHStreamingSessionURL. This action returns a URL (containing a session token) for accessing the MPEG-DASH session. You can then use the URL in a media player or a standalone application to display the stream.

An Amazon Kinesis video stream has the following requirements for providing video through MPEG-DASH:

- For streaming video playback track requirements, see the section called “GetDASHStreamingSessionURL”.
- Data retention must be greater than 0.
- The video track of each fragment must contain codec private data in the Advanced Video Coding (AVC) for H.264 format and HEVC for H.265 format. For more information, see MPEG-4 specification ISO/IEC 14496-15. For information about adapting stream data to a given format, see NAL Adaptation Flags.
- The audio track (if present) of each fragment must contain codec private data in the AAC format (AAC specification ISO/IEC 13818-7) or the MS Wave format.

Example: Using MPEG-DASH in HTML and JavaScript

The following example shows how to retrieve an MPEG-DASH streaming session for a Kinesis video stream and play it back in a webpage. The example shows how to play back video in the following players:

- Google Shaka Player
- dash.js

Topics

- Set Up the Kinesis Video Streams Client for MPEG-DASH Playback
- Retrieve the Kinesis Video Streams Archived Content Endpoint for MPEG-DASH Playback
- Retrieve the MPEG-DASH Streaming Session URL
- Display the Streaming Video with MPEG-DASH Playback
- Completed Example

Set Up the Kinesis Video Streams Client for MPEG-DASH Playback

To access streaming video with MPEG-DASH, first create and configure the Kinesis Video Streams client (to retrieve the service endpoint) and archived media client (to retrieve the MPEG-DASH streaming session). The application retrieves the necessary values from input boxes on the HTML page.

```
var streamName = $('#streamName').val();

// Step 1: Configure SDK Clients
var options = {
    accessKeyId: $('#accessKeyId').val(),
    secretAccessKey: $('#secretAccessKey').val(),
    sessionToken: $('#sessionToken').val() || undefined,
    region: $('#region').val(),
    endpoint: $('#endpoint').val() || undefined
    }
var kinesisVideo = new AWS.KinesisVideo(options);
var kinesisVideoArchivedContent = new AWS.KinesisVideoArchivedMedia(options);
```

Retrieve the Kinesis Video Streams Archived Content Endpoint for MPEG-DASH Playback

After the clients are initiated, retrieve the Kinesis Video Streams archived content endpoint so that you can retrieve the MPEG-DASH streaming session URL as follows:

```
 // Step 2: Get a data endpoint for the stream
console.log('Fetching data endpoint');
kinesisVideo.getDataEndpoint({
    StreamName: streamName,
    APIName: "GET_DASH_STREAMING_SESSION_URL"
}, function(err, response) {
    if (err) { return console.error(err); }
    console.log('Data endpoint: ' + response.DataEndpoint);
    kinesisVideoArchivedContent.endpoint = new AWS.Endpoint(response.DataEndpoint);
```

Retrieve the MPEG-DASH Streaming Session URL

When you have the archived content endpoint, call the [GetDASHStreamingSessionURL](https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/API_reader_GetDASHStreamingSessionURL.html) API to retrieve the MPEG-DASH streaming session URL as follows:

```
// Step 3: Get a Streaming Session URL
var consoleInfo = 'Fetching ' + protocol + ' Streaming Session URL';
console.log(consoleInfo);
```

```
if (protocol === 'DASH') {
    kinesisVideoArchivedContent.getDASHStreamingSessionURL({
        StreamName: streamName,
        PlaybackMode: $('#playbackMode').val(),
        DASHFragmentSelector: {
            FragmentSelectorType: $('#fragmentSelectorType').val(),
            TimestampRange: $('#playbackMode').val() === "LIVE" ? undefined : {
                StartTimestamp: new Date($('#startTimestamp').val()),
                EndTimestamp: new Date($('#endTimestamp').val())
            }
        },
        DisplayFragmentTimestamp: $('#displayFragmentTimestamp').val(),
        DisplayFragmentNumber: $('#displayFragmentNumber').val(),
        MaxManifestFragmentResults: parseInt($('#maxResults').val()),
        Expires: parseInt($('#expires').val())
    }, function(err, response) {
        if (err) { return console.error(err); }
        console.log('DASH Streaming Session URL: ' + response.DASHStreamingSessionURL);
```

Display the Streaming Video with MPEG-DASH Playback

When you have the MPEG-DASH streaming session URL, provide it to the video player. The method for providing the URL to the video player is specific to the player that you use.

The following code example shows how to provide the streaming session URL to a Google Shaka player:

```
// Step 4: Give the URL to the video player.

//Shaka Player elements
<video id="shaka" class="player" controls autoplay></video>
<script src="https://cdnjs.cloudflare.com/ajax/libs/shaka-player/2.4.1/shaka-
player.compiled.js">
</script>
...

var playerName = $('#player').val();

if (playerName === 'Shaka Player') {
    var playerElement = $('#shaka');
```

```
    playerElement.show();

    var player = new shaka.Player(playerElement[0]);
    console.log('Created Shaka Player');

    player.load(response.DASHStreamingSessionURL).then(function() {
        console.log('Starting playback');
    });
    console.log('Set player source');
}
```

The following code example shows how to provide the streaming session URL to an dash.js player:

```
<!-- dash.js Player elements -->
<video id="dashjs" class="player" controls autoplay=""></video>
<script src="https://cdn.dashjs.org/latest/dash.all.min.js"></script>

...

var playerElement = $('#dashjs');
playerElement.show();

var player = dashjs.MediaPlayer().create();
console.log('Created DASH.js Player');

player.initialize(document.querySelector('#dashjs'), response.DASHStreamingSessionURL,
 true);
console.log('Starting playback');
console.log('Set player source');
}
```

Completed Example

You can download or view the completed example code on GitHub.

Using streaming metadata with Kinesis Video Streams

You can use the Amazon Kinesis Video Streams Producer SDK to embed metadata at the individual fragment level in a Kinesis video stream. Metadata in Kinesis Video Streams is a mutable key-value pair. You can use it to describe the content of the fragment, embed associated sensor readings that must be transferred along with the actual fragment, or meet other custom needs. The metadata is made available as part of the the section called "GetMedia" or the section called "GetMediaForFragmentList" API operations. It's stored along with the fragments for the entire duration of the stream's retention period. Your consuming applications can read, process, and react based on the metadata using the Kinesis video stream Parser Library.

There are two modes in which the metadata can be embedded with fragments in a stream:

- **Nonpersistent** – You can affix metadata on a one-time, or ad hoc basis to fragments in a stream, based on business-specific criteria that have occurred. An example is a smart camera that detects motion and adds metadata to the corresponding fragments that contain the motion before sending the fragments to its Kinesis video stream. You might apply metadata to the fragment in the following format: `Motion = true`.
- **Persistent** – You can affix metadata to successive, consecutive fragments in a stream based on a continuing need. An example is a smart camera that sends the current latitude and longitude coordinates associated with all fragments that it sends to its Kinesis video stream. You might apply metadata to all the fragments in the following format: `Lat = 47.608013N , Long = -122.335167W`

You can affix metadata in both of these modes to the same fragment simultaneously, based on your application's needs. The embedded metadata might include objects detected, activity tracked, GPS coordinates, or any other custom data that you want to associate with the fragments in the stream. Metadata is encoded as key-value string pairs.

Topics

Adding metadata to a Kinesis video stream

Metadata that you add to a Kinesis video stream is modeled as MKV tags, which are implemented as key-value pairs.

Metadata can either be *transient*, such as to mark an event within the stream, or *persistent*, such as to identify fragments where a given event is taking place. A persistent metadata item remains, and is applied to each consecutive fragment, until it's canceled.

> **Note**
>
> The metadata items added using the Producer Libraries are distinct from the stream-level tagging APIs implemented with the section called “TagStream”, the section called “UntagStream”, and the section called “ListTagsForStream”.

Streaming metadata API

You can use the following operations in the Producer SDK to implement streaming metadata.

PIC

```
PUBLIC_API STATUS putKinesisVideoFragmentMetadata(STREAM_HANDLE streamHandle,
    PCHAR name,
    PCHAR value,
    BOOL persistent);
```

C++ Producer SDK

```
/**
 * Appends a "tag" or metadata - a key/value string pair into the stream.
 */
bool putFragmentMetadata(const std::string& name, const std::string& value, bool
 persistent = true);
```

Java Producer SDK

You can use the Java Producer SDK, to add metadata to a `MediaSource` using `MediaSourceSink.onCodecPrivateData`:

```
void onFragmentMetadata(final @Nonnull String metadataName, final @Nonnull String
 metadataValue, final boolean persistent)
throws KinesisVideoException;
```

Persistent and nonpersistent metadata

For nonpersistent metadata, you can add multiple metadata items with the same *name*. The Producer SDK collects the metadata items in the metadata queue until they are prepended to the next fragment. The metadata queue is cleared as the metadata items are applied to the stream. To repeat the metadata, call `putKinesisVideoFragmentMetadata` or `putFragmentMetadata` again.

For persistent metadata, the Producer SDK collects the metadata items in the metadata queue in the same way as for nonpersistent metadata. However, the metadata items aren't removed from the queue when they are prepended to the next fragment.

Calling `putKinesisVideoFragmentMetadata` or `putFragmentMetadata` with `persistent` set to `true` has the following behavior:

- Calling the API puts the metadata item in the queue. The metadata is added as an MKV tag to every fragment while the item is in the queue.
- Calling the API with the same *name* and a different *value* as a previously added metadata item overwrites the item.
- Calling the API with an empty *value* removes (cancels) the metadata item from the metadata queue.

Consuming metadata embedded in a Kinesis video stream

To consume the metadata in a Kinesis video stream, use an implementation of `MkvTagProcessor`:

```
public interface MkvTagProcessor {
        default void process(MkvTag mkvTag, Optional<FragmentMetadata>
 currentFragmentMetadata) {
            throw new NotImplementedException("Default
 FragmentMetadataVisitor.MkvTagProcessor");
        }
        default void clear() {
            throw new NotImplementedException("Default
 FragmentMetadataVisitor.MkvTagProcessor");
     }
    }
}
```

This interface is found in the [FragmentMetadataVisitor](#) class in the [Kinesis video stream Parser Library](#).

The `FragmentMetadataVisitor` class contains an implementation of `MkvTagProcessor`:

```
public static final class BasicMkvTagProcessor implements
 FragmentMetadataVisitor.MkvTagProcessor {
    @Getter
    private List<MkvTag> tags = new ArrayList<>();

    @Override
    public void process(MkvTag mkvTag, Optional<FragmentMetadata>
 currentFragmentMetadata) {
        tags.add(mkvTag);
    }

    @Override
    public void clear() {
        tags.clear();
 }
}
```

The `KinesisVideoRendererExample` class contains an example of how to use a `BasicMkvTagProcessor`. In the following example, a `BasicMkvTagProcessor` is added to the `MediaProcessingArguments` of an application:

```
if (renderFragmentMetadata) {
    getMediaProcessingArguments =
 KinesisVideoRendererExample.GetMediaProcessingArguments.create(
        Optional.of(new FragmentMetadataVisitor.BasicMkvTagProcessor()));
```

The `BasicMkvTagProcessor.process` method is called when fragment metadata arrives. You can retrieve the accumulated metadata with `GetTags`. To retrieve a single metadata item, first call `clear` to clear the collected metadata, and then retrieve the metadata items again.

Streaming metadata limits

See the section called "Fragment metadata quotas" for more information about the limits that apply to adding streaming metadata to a Kinesis video stream

Kinesis Video Streams data model

The Producer Libraries and Stream Parser Library send and receive video data in a format that supports embedding information alongside video data. This format is based on the Matroska (MKV) specification.

The MKV format is an open specification for media data. All the libraries and code examples in the *Amazon Kinesis Video Streams Developer Guide* send or receive data in the MKV format.

The Kinesis Video Streams Producer Libraries uses the `StreamDefinition` and `Frame` types to produce MKV stream headers, frame headers, and frame data.

For information about the full MKV specification, see Matroska Specifications.

The following sections describe the components of MKV-formatted data produced by the C++ Producer Library.

Topics

- Stream header elements
- Stream track data
- Frame header elements
- MKV frame data

Stream header elements

The following MKV header elements are used by `StreamDefinition` (defined in `StreamDefinition.h`).

Element	Description	Typical values
stream_name	Corresponds to the name of the Kinesis video stream.	*my-stream*
retention_period	The duration, **in hours**, that stream data is persisted by Kinesis Video Streams. Specify `0` for a stream that doesn't retain data.	*24*
tags	A key-value collection of user data. This data is displayed in the AWS Management Console and can be read by client applications to filter	

Element	Description	Typical values
	or get information about a stream.	
kms_key_id	If present, the user-defined AWS KMS key is used to encrypt data on the stream. If absent, the data is encrypted by the Kinesis-supplied key (`aws/kinesis-video`).	*01234567-89ab-cdef-0123-456789ab*
streaming_type	Currently, the only valid streaming type is `STREAMING_TYPE_REALTIME` .	*STREAMING_TYPE_REALTIME*
content_type	The user-defined content type. For streaming video data to play in the console, the content type must be `video/h264` .	*video/h264*
max_latency	This value isn't currently used and should be set to 0.	*0*
fragment_duration	The estimate of how long your fragments should be, which is used for optimization. The actual fragment duration is determined by the streaming data.	*2*

Element	Description	Typical values
timecode_scale	Indicates the scale used by frame timestamps. The default is 1 millisecond. Specifying `0` also assigns the default value of 1 milliseco nd. This value can be between 100 nanoseconds and 1 second. For more information, see TimecodeScale in the Matroska documentation.	
key_frame_fragmentation	If `true`, the stream starts a new cluster when a keyframe is received.	*true*
frame_timecodes	If `true`, Kinesis Video Streams uses the presentat ion time stamp (pts) and decoding time stamp (dts) values of the received frames. If `false`, Kinesis Video Streams stamps the frames when they are received with system-generated time values.	*true*

Element	Description	Typical values
absolute_fragment_time	If `true`, the cluster timecodes are interpreted as using absolute time (for example, from the producer's system clock). If `false`, the cluster timecodes are interpreted as being relative to the start time of the stream.	*true*
fragment_acks	If `true`, acknowledgements (ACKs) are sent when Kinesis Video Streams receives the data. The ACKs can be received using the `KinesisVideoStreamFragmentAck` or `KinesisVideoStreamParseFragmentAck` callbacks.	*true*
restart_on_error	Indicates whether the stream should resume transmission after a stream error is raised.	*true*
nal_adaptation_flags	Indicates whether NAL (Network Abstraction Layer) adaptation or codec private data is present in the content. Valid flags include `NAL_ADAPTATION_ANNEXB_NALS` and `NAL_ADAPTATION_ANNEXB_CPD_NALS`.	*NAL_ADAPTATION_ANNEXB_NALS*

Element	Description	Typical values
frame_rate	An estimate of the content frame rate. This value is used for optimization; the actual frame rate is determined by the rate of incoming data. Specifying `0` assigns the default of 24.	*24*
avg_bandwidth_bps	An estimate of the content bandwidth, **in Mbps**. This value is used for optimization; the actual rate is determined by the bandwidth of incoming data. For example, for a 720 p resolution video stream running at 25 FPS, you can expect the average bandwidth to be 5 Mbps.	*5*
buffer_duration	The duration that content is to be buffered on the producer. If there's low network latency, this value can be reduced. If network latency is high, increasing this value prevents frames from being dropped before they can be sent, due to allocation failing to put frames into the smaller buffer.	

Element	Description	Typical values
replay_duration	The amount of time the video data stream is "rewound" if connection is lost. This value can be zero if lost frames due to connection loss are not a concern. The value can be increased if the consuming application can remove redundant frames. This value should be less than the buffer duration, otherwise the buffer duration is used.	
connection_staleness	The duration that a connectio n is maintained when no data is received.	
codec_id	The codec used by the content. For more informati on, see CodecID in the Matroska specification.	*V_MPEG2*
track_name	The user-defined name of the track.	*my_track*

Element	Description	Typical values
codecPrivateData	Data provided by the encoder used to decode the frame data, such as the frame width and height in pixels, which is needed by many downstream consumers. In the C++ Producer Library, the `gMkvTrackVideoBits` array in `MkvStatics.cpp` includes pixel width and height for the frame.	
codecPrivateDataSize	The size of the data in the `codecPrivateData` parameter.	
track_type	The type of the track for the stream.	*MKV_TRACK_INFO_TYPE_AUDIO* or *MKV_TRACK_INFO_TYPE_VIDEO*
segment_uuid	User-defined segment uuid (16 bytes).	
default_track_id	Unique non-zero number for the track.	*1*

Stream track data

The following MKV track elements are used by `StreamDefinition` (defined in `StreamDefinition.h`).

Element	Description	Typical Values
track_name	User-defined track name. For example, "audio" for the audio track.	*audio*
codec_id	Codec id for the track. For example, "A_AAC" for an audio track.	*A_AAC*
cpd	Data provided by the encoder used to decode the frame data. This data can include frame width and height in pixels, which is needed by many downstream consumers . In the C++ Producer Library, the gMkvTrackVideoBits array in MkvStatics.cpp includes pixel width and height for the frame.	
cpd_size	The size of the data in the codecPrivateData parameter.	
track_type	The type of the track. For example, you can use the enum value of MKV_TRACK _INFO_TYPE_AUDIO for audio.	*MKV_TRACK_INFO_TYP E_AUDIO*

Frame header elements

The following MKV header elements are used by `Frame` (defined in the `KinesisVideoPic` package, in `mkvgen/Include.h`):

- **Frame Index:** A monotonically increasing value.

- **Flags:** The type of frame. Valid values include the following:
  - `FRAME_FLAGS_NONE`
  - `FRAME_FLAG_KEY_FRAME`: If `key_frame_fragmentation` is set on the stream, key frames start a new fragment.
  - `FRAME_FLAG_DISCARDABLE_FRAME`: Tells the decoder that it can discard this frame if decoding is slow.
  - `FRAME_FLAG_INVISIBLE_FRAME`: Duration of this block is 0.
- **Decoding Timestamp:** The timestamp of when this frame was decoded. If previous frames depend on this frame for decoding, this timestamp might be earlier than that of earlier frames. This value is relative to the start of the fragment.
- **Presentation Timestamp:** The timestamp of when this frame is displayed. This value is relative to the start of the fragment.
- **Duration:** The playback duration of the frame.
- **Size:** The size of the frame data in bytes

MKV frame data

The data in `frame.frameData` might contain only media data for the frame, or it might contain further nested header information, depending on the encoding schema used. To be displayed in the AWS Management Console, the data must be encoded in the H.264 codec, but Kinesis Video Streams can receive time-serialized data streams in any format.

Getting started with Amazon Kinesis Video Streams

This section describes how to perform the following tasks in Amazon Kinesis Video Streams:

- Set up your AWS account and create an administrator, if you haven't already done so.
- Create a Kinesis video stream.
- Send data to the Kinesis video stream from your camera and view the media in the console.

If you're new to Amazon Kinesis Video Streams, we recommend that you read Kinesis Video Streams: How it works first.

> **Note**
>
> Following the Getting started sample will not incur any charges to your AWS account. For data costs in your Region, see Amazon Kinesis Video Streams Pricing.

Topics

Set up an account

Before you use Amazon Kinesis Video Streams for the first time, complete the following tasks.

Topics

Sign up for an AWS account

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

To sign up for an AWS account

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

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

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

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

Create a user with administrative access

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

Secure your AWS account root user

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

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.
2. Turn on multi-factor authentication (MFA) for your root user.

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

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.
2. In IAM Identity Center, grant administrative access to a user.

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

Sign in as the user with administrative access

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

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

Assign access to additional users

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

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

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

Create an AWS account key

You will need an AWS account Key to access Amazon Kinesis Video Streams programmatically.

To create an AWS account Key, do the following:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. Choose **Users** in the navigation bar, and choose the **Administrator** user.
3. Choose the **Security credentials** tab, and choose **Create access key**.
4. Record the **Access key ID**. Choose **Show** under **Secret access key**. Record the **Secret access key**.

Create a Kinesis video stream

This section describes how to create a Kinesis video stream.

This section contains the following procedures:

- the section called "Create a video stream using the console"
- the section called "Create a video stream using the AWS CLI"

Create a video stream using the console

1. Open the console at https://console.aws.amazon.com/kinesisvideo/home.
2. On the **Video streams** page, choose **Create video stream**.
3. On the **Create a new video stream** page, enter *YourStreamName* for the stream name. Leave the **Default configuration** button selected.
4. Choose **Create video stream**.
5. After Amazon Kinesis Video Streams creates the stream, review the details on the **YourStreamName** page.

Create a video stream using the AWS CLI

1. Verify that you have the AWS CLI installed and configured. For more information, see the AWS Command Line Interface documentation.
2. Run the following `Create-Stream` command in the AWS CLI:

```
aws kinesisvideo create-stream --stream-name "YourStreamName" --data-retention-in-hours 24
```

The response will look similar to the following:

```
{
    "StreamARN": "arn:aws:kinesisvideo:us-
west-2:123456789012:stream/YourStreamName/123456789012"
}
```

Send data to an Amazon Kinesis video stream

This section describes how to send media data from a camera to the Kinesis video stream that you created in the previous section. This section uses the C++ Producer Library as a GStreamer Plugin - kvssink plugin.

To send media from a variety of devices on a variety of operating systems, this tutorial uses the Kinesis Video Streams C++ producer library and GStreamer, an open-source media framework that standardizes access to cameras and other media sources.

Topics

Build the SDK and samples

You can build the SDK and samples on your computer or in AWS Cloud9. Follow the appropriate procedures below.

Build on your computer

Use the instructions in the readme file to build the producer library and sample application.

This includes:

- Installing dependencies
- Cloning the repository
- Using **CMake** to generate **makefiles**
- Building the binary files using **make**

Build in AWS Cloud9

Follow these procedures to upload to Kinesis Video Streams in AWS Cloud9. You won't need to download anything to your computer.

1. In the AWS Management Console, open AWS Cloud9.

   Select **Create environment**.

2. On the **Create environment** screen, complete the following:

   - **Name** - Type a name for your new environment.
   - **Platform** - Select **Ubuntu Server 22.04 LTS**.

   You can leave the other fields with the default selections.

3. When the environment has been created, select **Open** in the **Cloud9 IDE** column.

   In the lower-middle area of the screen, you see `Admin:~/environment $`. This is the AWS Cloud9 (Amazon EC2) terminal.

   > **Note**
   >
   > If you accidentally close the terminal, select **Window**, **New Terminal**.

   Run the following commands in the terminal to change the volume to 20 GiB.

   a. Download the script.

   ```
   wget https://awsj-iot-handson.s3-ap-northeast-1.amazonaws.com/kvs-workshop/
   resize_volume.sh
   ```

   b. Give the script execute permissions.

   ```
   chmod +x resize_volume.sh
   ```

   c. Run the script.

   ```
   ./resize_volume.sh
   ```

4. Fetch the latest information on all of the software you can install or update through the Advanced Packaging Tool (APT).

   This command doesn't update the software itself, but makes sure your system knows what the latest available versions are.

   ```
   sudo apt-get update
   ```

5. Install the C++ producer SDK dependencies.

   ```
   sudo apt-get install -y cmake m4 git build-essential pkg-config libssl-dev
    libcurl4-openssl-dev \
   liblog4cplus-dev libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev \
   gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-bad gstreamer1.0-plugins-
   good \
   gstreamer1.0-plugins-ugly gstreamer1.0-tools
   ```

6. Use git to clone the C++ producer SDK.

   ```
   git clone https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-
   cpp.git
   ```

7. Prepare a build directory.

   ```
   cd amazon-kinesis-video-streams-producer-sdk-cpp
   mkdir build
   cd build
   ```

8. Use CMake to generate makefiles.

   ```
   cmake .. -DBUILD_GSTREAMER_PLUGIN=TRUE -DBUILD_DEPENDENCIES=OFF
   ```

   The end of the expected output looks like the following:

   ```
   -- Build files have been written to: /home/ubuntu/environment/amazon-kinesis-
   video-streams-producer-sdk-cpp/build
   ```

9. Use make to compile the SDK and sample applications, as well as build the final executables.

```
make
```

The end of the expected output looks like the following:

```
[100%] Linking CXX executable kvs_gstreamer_file_uploader_sample
[100%] Built target kvs_gstreamer_file_uploader_sample
```

10. Confirm the sample files were built. List the files in the current directory:

    ```
    ls
    ```

    Confirm that the following files are present:

    - kvs_gstreamer_sample
    - libgstkvssink.so

11. (Optional) You can add setting the GST_PLUGIN_PATH environment variable to your shell's start-up script. This ensures GST_PLUGIN_PATH is set properly during a new terminal session. In AWS Cloud9, the shell's start-up script is: `~/.bashrc`.

    Run the following command to append the command to the end of the shell's start-up script.

    ```
    echo "export GST_PLUGIN_PATH=~/environment/amazon-kinesis-video-streams-
    producer-sdk-cpp/build" >> ~/.bashrc
    ```

    Type the following to run the shell's start-up script:

    ```
    source ~/.bashrc
    ```

    Confirm GST_PLUGIN_PATH is set.

    ```
    echo $GST_PLUGIN_PATH
    ```

    If you set the output correctly, you will see the following output. If the output is blank, the environment variable is not set properly.

```
/home/ubuntu/environment/amazon-kinesis-video-streams-producer-sdk-cpp/build
```

Run the samples to upload media to Kinesis Video Streams

The sample application does not support IMDS credentials. In your terminal, export AWS credentials for your IAM user or role and the region your stream is located in.

```
export AWS_ACCESS_KEY_ID=YourAccessKey
export AWS_SECRET_ACCESS_KEY=YourSecretKey
export AWS_DEFAULT_REGION=YourAWSRegion
```

If you're using temporary AWS credentials, also export your session token:

```
export AWS_SESSION_TOKEN=YourSessionToken
```

.mp4 files

Download a sample .mp4 video to upload to Kinesis Video Streams.

```
wget https://awsj-iot-handson.s3-ap-northeast-1.amazonaws.com/kvs-workshop/
sample.mp4
```

Video specifications:

- **Resolution** - 1280 x 720 pixels
- **Frame rate** - 30 frames per second
- **Duration** - 14.0 seconds
- **Video encoding** - H.264, in track 1
- **Keyframes** - Every 3 seconds, resulting in a fragment duration (also known as a group of pictures (GoP) size) of 3 seconds, with the final fragment being 2 seconds long.

Run the following command with the name of the stream you previously created. If you haven't created a stream yet, see the section called "Create a Kinesis video stream".

```
./kvs_gstreamer_sample YourStreamName ./sample.mp4
```

Sample video from GStreamer

Use the following command to generate a video using GStreamer.

Tell GStreamer where to locate the `kvssink` GStreamer plugin. In your build directory, specify the path to the folder containing the `libgstkvssink.so` file.

From your build directory, run the following command:

```
export GST_PLUGIN_PATH=`pwd`
```

This GStreamer pipeline generates a live test video stream with a standard test pattern that runs at 10 frames per second with a resolution of 640x480 pixels. An overlay is added displaying the current system time and date. The video is then encoded into H.264 format and keyframes are generated at most every 10 frames, resulting in a fragment duration (also known as a group of pictures (GoP) size) of 1 second. `kvssink` takes the H.264-encoded video stream, packages it into the Matroska (MKV) container format, and uploads it to your Kinesis video stream.

Run the following command:

```
gst-launch-1.0 -v videotestsrc is-live=true \
  ! video/x-raw,framerate=10/1,width=640,height=480 \
  ! clockoverlay time-format="%a %B %d, %Y %I:%M:%S %p" \
  ! x264enc bframes=0 key-int-max=10 \
  ! h264parse \
  ! kvssink stream-name="YourStreamName"
```

To stop the GStreamer pipeline, select the terminal window and press **CTRL+C**.

> **Note**
>
> For more information about using the GStreamer plugin to stream video from an RTSP stream from a camera, or from a USB camera, see Example: Kinesis Video Streams Producer SDK GStreamer Plugin - kvssink.

Review acknowledgement objects

During upload, Kinesis Video Streams will send acknowledgement objects back to the client performing the upload. You should see these printed in the command output. An example looks like the following:

```
{"EventType":"PERSISTED","FragmentTimecode":1711124585823,"FragmentNumber":"12345678901234567890
```

If the acknowledgement's `EventType` is `PERSISTED`, it means Kinesis Video Streams has durably stored and encrypted this chunk of media for retrieval, analysis, and long-term storage.

For more information about acknowledgements, see the section called "PutMedia".

Consume media data

You can consume media data by either viewing it in the console, or by creating an application that reads media data from a stream using Hypertext Live Streaming (HLS).

View media in the console

In another browser tab, open the AWS Management Console. In the Kinesis Video Streams Dashboard, select Video streams.

Select the name of your stream in the list of streams. Use the search bar, if necessary.

Expand the **Media playback** section. If the video is still uploading, it will be shown. If the upload has finished, select the double-left arrow.

Consume media data using HLS

You can create a client application that consumes data from a Kinesis video stream using HLS. For information about creating an application that consumes media data using HLS, see the section called "Video Playback".

Amazon Kinesis Video Streams Edge Agent

Amazon Kinesis Video Streams offers an efficient, cost-effective way to connect to IP cameras on customer premises. With the Amazon Kinesis Video Streams Edge Agent, you can locally record and store video from the cameras and stream videos to the cloud on a customer-defined schedule for long-term storage, playback, and analytical processing.

> **Note**
>
> To access the Amazon Kinesis Video Streams Edge Agent, complete this brief form.

You can download the Amazon Kinesis Video Streams Edge Agent and deploy it at your on-premises edge compute devices. You can also easily deploy them in Docker containers running on Amazon EC2 instances. After deployment, you can use the Amazon Kinesis Video Streams API to update video recording and cloud uploading configurations. The feature works with any IP camera that can stream over RTSP protocol. It doesn't require any additional firmware deployment to the cameras.

We offer the following installations for the Amazon Kinesis Video Streams Edge Agent:

- **As an AWS IoT Greengrass V2 component:** You can install the Amazon Kinesis Video Streams Edge Agent as an AWS IoT Greengrass component on any AWS IoT Greengrass certified device. To learn more about AWS IoT Greengrass, see the AWS IoT Greengrass Version 2 Developer Guide.
- **On AWS Snowball Edge:** You can run the Amazon Kinesis Video Streams Edge Agent on Snowball Edge devices. To learn more, see the AWS Snowball Edge Developer Guide.
- **On a native AWS IoT deployment:** You can install the Amazon Kinesis Video Streams Edge Agent natively on any compute instance. Edge SDK uses AWS IoT Core for managing edge through the the section called "Amazon Kinesis Video Streams".

To get started with Amazon Kinesis Video Streams Edge Agent, continue with the appropriate procedures below.

Topics

- Monitoring Amazon Kinesis Video Streams Edge Agent
- Run Amazon Kinesis Video Streams Edge Agent in non-AWS IoT Greengrass mode
- Deploy the Amazon Kinesis Video Streams Edge Agent to AWS IoT Greengrass
- Amazon Kinesis Video Streams Edge Agent FAQ

Amazon Kinesis Video Streams Edge Agent API operations

Use the following API operations to configure the Amazon Kinesis Video Streams Edge Agent:

- the section called "StartEdgeConfigurationUpdate"
- the section called "DescribeEdgeConfiguration"
- the section called "DeleteEdgeConfiguration"
- the section called "ListEdgeAgentConfigurations"

Monitoring Amazon Kinesis Video Streams Edge Agent

To monitor your Amazon Kinesis Video Streams Edge Agent, see the section called "Monitoring the Amazon Kinesis Video Streams Edge Agent with CloudWatch".

Run Amazon Kinesis Video Streams Edge Agent in non-AWS IoT Greengrass mode

Follow these steps to run the Amazon Kinesis Video Streams Edge Agent with AWS IoT MQTT as a standalone deployment.

Topics

- Step 1: Install the necessary dependencies on the device
- Step 2: Create the Amazon Kinesis Video Streams and AWS Secrets Manager resources for your IP camera RTSP URLs
- Step 3: Create an IAM permissions policy
- Step 4: Create an IAM role
- Step 5: Create the AWS IoT role alias
- Step 6: Create the AWS IoT policy

Step 1: Install the necessary dependencies on the device

> **Note**
>
> For a list of supported operating systems, see the section called “What operating systems does Amazon Kinesis Video Streams Edge Agent support?”.

Install dependencies on the device

1. To run the Amazon Kinesis Video Streams Edge Agent, install the following appropriate libraries on your device:

   Ubuntu

   Type:

   ```
   wget -O- https://apt.corretto.aws/corretto.key | sudo apt-key add -
   sudo add-apt-repository 'deb https://apt.corretto.aws stable main'
   sudo apt-get update

   sudo apt-get install -y gcc libssl-dev libcurl4-openssl-dev liblog4cplus-dev \
   libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev \
   gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-bad \
   gstreamer1.0-plugins-good gstreamer1.0-tools \
   unzip java-11-amazon-corretto-jdk maven
   ```

   Amazon Linux 2

   Type:

   ```
   sudo yum update -y && sudo yum upgrade -y && sudo yum clean all -y
   sudo yum install -y gcc-c++ openssl-devel libcurl-devel gstreamer1* wget \
   java-11-amazon-corretto tar
   ```

Install `log4cplus-2.1.0` from the source.

```
wget https://github.com/log4cplus/log4cplus/releases/download/REL_2_1_0/
log4cplus-2.1.0.tar.gz
tar -xzvf log4cplus-2.1.0.tar.gz
cd log4cplus-2.1.0 && \
mkdir build && \
cd build && \
cmake .. && \
sudo make && \
sudo make install
```

Install `apache-maven-3.9.2` from the source.

```
wget https://dlcdn.apache.org/maven/maven-3/3.9.2/binaries/apache-maven-3.9.2-
bin.tar.gz
RUN tar -xzvf apache-maven-3.9.2-bin.tar.gz -C /opt
```

> **Important**
>
> If you see a screen telling you that some services need to be restarted, press Enter to select **Ok**.

For additional information, see *Amazon Corretto 11 User Guide*.

2. Install the AWS Command Line Interface. See the Installing or updating the latest version of the AWS CLI procedures in the *AWS Command Line Interface User Guide*.

Step 2: Create the Amazon Kinesis Video Streams and AWS Secrets Manager resources for your IP camera RTSP URLs

Follow these procedures to create the streams and secrets needed in AWS Secrets Manager. Do this step first, because you need the ARNs of the created resources in the policies.

Create Amazon Kinesis Video Streams

Create Amazon Kinesis Video Streams using the AWS Management Console, AWS CLI, or API.

In the AWS Management Console, open the [Amazon Kinesis Video Streams console](). Choose **Video streams** in the left navigation.

For more information, see the section called “Create a Kinesis video stream”.

Create secrets in AWS Secrets Manager

In the AWS Management Console, open the AWS Secrets Manager console. Choose **Secrets** in the left navigation.

Verify that the appropriate Region is selected.

1. Choose **Store a new secret**.
   a. **Step 1: Choose secret type**
      - Select **Other type of secret**.
      - In the **Key/Value Pairs** section, add a key-value pair.

        Key: `MediaURI`

        > **Note**
        >
        > The key must be `MediaURI`. This is case-sensitive. If you enter it incorrectly, the application doesn't work.

        Value: *`Your MediaURI`*.

        Example

        Example: `rtsp://<YourCameraIPAddress>:<YourCameraRTSPPort>/YourCameraMediaURI`.
   b. **Step 2: Configure secret**. Give this secret a name. Name it whatever you want.
   c. **Step 3: Configure rotation - optional**. Choose **Next**.
   d. **Step 4: Review**. Choose **Store**.
2. If your secret does not display immediately, select the refresh button.

   Choose the name of your secret. Make note of the **Secret ARN**.
3. Repeat this process for each MediaURI that you want to stream from.

> **Note**
>
> The AWS network blocks some public RTSP sources. You cannot access these from within the Amazon EC2 instance or if you are running unmanaged while connected to the VPN.
>
> > **Important**
> >
> > Your camera RTSP URL should stream video in h.264 format. The fragment duration must not exceed the limit mentioned in the section called "Producer SDK limits". Amazon Kinesis Video Streams Edge Agent only supports video.
>
> Run `gst-discoverer-1.0 Your RtspUrl` to make sure that your camera is reachable from your device.

Save the ARNs for all of the streams and secrets that you created. You need these for the next step.

Step 3: Create an IAM permissions policy

Follow these procedures to create an IAM policy. This permissions policy allows selective access control (a subset of supported operations) for an AWS resource. In this case, the AWS resources are the video streams that you want the Amazon Kinesis Video Streams Edge Agent to stream to. The resources also include the AWS Secrets Manager secrets that the Amazon Kinesis Video Streams Edge Agent can retrieve. For more information, see IAM policies.

Create a policy by using the JSON policy editor

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policies**.

   If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Choose **Get Started**.
3. At the top of the page, choose **Create policy**.
4. In the **Policy editor** section, choose the **JSON** option.
5. Enter the following JSON policy document:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "kinesisvideo:ListStreams",
                "iot:Connect",
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:DescribeStream",
                "kinesisvideo:PutMedia",
                "kinesisvideo:TagStream",
                "kinesisvideo:GetDataEndpoint"
            ],
             "Resource": [
                "arn:aws:kinesisvideo:*:*:stream/streamName1/*",
                "arn:aws:kinesisvideo:*:*:stream/streamName2/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                 "arn:aws:secretsmanager:*:*:secret:*",
                 "arn:aws:secretsmanager:*:*:secret:*"
            ]
        }
    ]
}
```

> **Note**
>
> Replace `arn:aws:kinesisvideo:*:*:stream/streamName1/*` and `arn:aws:kinesisvideo:*:*:stream/streamName2/*` with the ARNs for the video streams, and replace `arn:aws:secretsmanager:*:*:secret:*` with the ARNs that contain the MediaURI secrets that you created in [the section called "2. Create resources for your IP camera RTSP URLs"](). Use the ARNs for the secrets that you want the Amazon Kinesis Video Streams Edge Agent to access.

6. Choose **Next**.

> **Note**
>
> You can switch between the **Visual** and **JSON** editor options anytime. However, if you make changes or choose **Next** in the **Visual** editor, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy restructuring]() in the IAM User Guide.

7. On the **Review and create** page, enter a **Policy name** and an optional **Description** for the policy that you are creating. Review **Permissions defined in this policy** to see the permissions that are granted by your policy.
8. Choose **Create policy** to save your new policy.

Step 4: Create an IAM role

The role that you create in this step can be assumed by AWS IoT in order to obtain temporary credentials from the AWS Security Token Service (AWS STS). This is done when performing credential authorization requests from the Amazon Kinesis Video Streams Edge Agent.

Create the service role for Amazon Kinesis Video Streams (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. Choose the **Custom trust policy** role type and paste the following policy:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "credentials.iot.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }
}
```

4. Select the box next to the IAM policy that you created in the section called “3. Create an IAM permissions policy”.
5. Choose **Next**.
6. Enter a role name or role name suffix to help you identify the purpose of this role.

   Example

   Example: `KvsEdgeAgentRole`
7. (Optional) For **Description**, enter a description for the new role.
8. (Optional) Add metadata to the role by attaching tags as key/value pairs.

   For more information about using tags in IAM, see Tagging IAM resources in the IAM User Guide.
9. Review the role and then choose **Create role**.

Step 5: Create the AWS IoT role alias

Follow these procedures to create an AWS IoT role alias for the IAM role that you created in the section called “4. Create an IAM role”. A role alias is an alternate data model that points to the IAM role. An AWS IoT credentials provider request must include a role alias to indicate which IAM role to assume in order to obtain temporary credentials from the AWS Security Token Service (AWS STS). For more information, see How to use a certificate to get a security token.

Create the AWS IoT role alias

1. Sign in to the AWS Management Console and open the AWS IoT Core console at https://console.aws.amazon.com/iot/.

2. Verify that the appropriate Region is selected.
3. On the left navigation, select **Security** and then choose **Role Aliases**.
4. Choose **Create role alias**.
5. Enter a name for your role alias.

   Example

   Example: `KvsEdgeAgentRoleAlias`
6. In the **Role** dropdown, select the IAM role you created in [the section called “4. Create an IAM role”](#).
7. Choose **Create**. On the next page, you see a note that your role alias was successfully created.
8. Search for and select the newly created role alias. Make note of the **Role alias ARN**. You need this for the AWS IoT policy in the next step.

Step 6: Create the AWS IoT policy

Follow these procedures to create an AWS IoT policy that will be attached to the device certificate. This gives permissions to AWS IoT capabilities and allows the assumption of the role alias using the certificate.

With AWS IoT Core policies, you can control access to the AWS IoT Core data plane. The AWS IoT Core data plane consists of operations that you can use to do the following:

- Connect to the AWS IoT Core message broker
- Send and receive MQTT messages
- Get or update a thing's device shadow

For more information, see AWS IoT Core policies.

Use AWS IoT policy editor to create an AWS IoT policy

1. Sign in to the AWS Management Console and open the AWS IoT Core console at https://console.aws.amazon.com/iot/.
2. On the left navigation, select **Security** and then choose **Policies**.
3. Choose **Create policy**.
4. Enter a name for your policy.

Example

An example of a policy name is **KvsEdgeAccessIoTPolicy**.

5. (Optional) Add metadata to the policy by attaching tags as key-value pairs.

   For more information about using tags in IAM, see Tagging your AWS IoT resources in the *AWS IoT Core Developer Guide*.

6. Choose the **JSON** tab.

7. Paste the following JSON policy document:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect",
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:AssumeRoleWithCertificate"
            ],
            "Resource": "your-role-alias-arn"
        }
    ]
}
```

> **Note**
>
> Replace `your-role-alias-arn` with the ARN of the role alias that you created in the section called "5. Create the AWS IoT role alias".

8. Choose **Create** to save your work.

Step 7: Create an AWS IoT thing and obtain credentials for AWS IoT Core

At this point you've created:

- An IAM permissions policy. See the section called “3. Create an IAM permissions policy”.
- An IAM role, with the permissions policy attached. See the section called “4. Create an IAM role”.
- An AWS IoT role alias for the IAM role. See the section called “5. Create the AWS IoT role alias”.
- An AWS IoT policy, currently unattached to any AWS resource. See the section called “6. Create the AWS IoT policy”.

To create and register an AWS IoT thing and get AWS IoT Core access credentials

1. Register the device as an AWS IoT thing and generate the X.509 certificate for the device.

   a. Sign in to the AWS Management Console and open the AWS IoT Core console at https://console.aws.amazon.com/iot/.

   b. Select the appropriate Region.

   c. On the left navigation, select **All devices**, then choose **Things**.

   d. Choose **Create things**.

   e. Select **Create single thing**, then choose **Next**.

      1. **Step 1. Specify thing properties**

         Type a name for your thing, then choose **Next**.

      2. **Step 2. Configure device certificate**

         Select **Auto-generate a new certificate (recommended)**, then choose **Next**.

      3. **Step 3. Attach policies to certificate**

         Search for the permissions policy you created in the section called “6. Create the AWS IoT policy”.

         Select the check box next to your policy and choose **Create thing**.

f. In the window that appears, download the following files:

- Device certificate. This is the X.509 certificate.
- Public key file
- Private key file
- Amazon trust services endpoint (RSA 2048 bit key: Amazon Root CA 1)

Make note of the location of each of these files for a later step.

g. Choose **Done**. On the next page, you see a note that your thing was successfully created.

h. Transfer the files downloaded above onto your AWS IoT thing, if not already there.

2. Obtain the credential provider endpoint for your AWS account.

AWS CLI

Run the following command:

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

AWS Management Console

In AWS CloudShell, run the following command:

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

Make note of this information for a later step.

3. Obtain the device data endpoint for your AWS account.

AWS CLI

Run the following command:

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

AWS Management Console

Do the following:

1. Sign in to the AWS Management Console and open the AWS IoT Core console at https://console.aws.amazon.com/iot/.
2. In the left navigation, select **Settings**.
3. Locate the **Device data endpoint**.

Make note of this information for a later step.

4. (Optional) Verify that your certificates were generated correctly.

Run the following command to validate that your items were generated correctly.

```
curl --header "x-amzn-iot-thingname:your-thing-name" \
  --cert /path/to/certificateID-certificate.pem.crt \
  --key /path/to/certificateID-private.pem.key \
  --cacert /path/to/AmazonRootCA1.pem \
  https://your-credential-provider-endpoint/role-aliases/your-role-alias-name/
credentials
```

For more information, see How to use a certificate to get a security token.

Step 8: Build and run the Amazon Kinesis Video Streams Edge Agent

Build and run the Amazon Kinesis Video Streams Edge Agent

1. Download the `tar` file using the link that was provided to you.

   If you completed the Amazon Kinesis Video Streams Edge Agent interest form, check your email for the download link. If you haven't completed the form, complete it here.

2. Verify the checksum.
3. Extract the binaries and jar in your device.

   Type: `tar -xvf kvs-edge-agent.tar.gz`.

   After extraction, your folder structure will look like the following:

```
kvs-edge-agent/LICENSE
kvs-edge-agent/THIRD-PARTY-LICENSES
kvs-edge-agent/pom.xml
```

```
kvs-edge-agent/KvsEdgeComponent
kvs-edge-agent/KvsEdgeComponent/recipes
kvs-edge-agent/KvsEdgeComponent/recipes/recipe.yaml
kvs-edge-agent/KvsEdgeComponent/artifacts
kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/edge_log_config
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/kvs-edge-agent.jar
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/libgstkvssink.so
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/libIngestorPipelineJNI.so
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib/libcproducer.so
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib/libKinesisVideoProducer.so
```

> **Note**
>
> The release folder name should be set up in a way that reflects the latest binary release number. For example, a 1.0.0 release will have the folder name set as 1.0.0.

4. Build the dependencies jar.

> **Note**
>
> The jar included with the `kvs-edge-agent.tar.gz` does not have the dependencies. Use the following steps to build those libraries.

Navigate to the `kvs-edge-agent` folder that contains `pom.xml`.

Type `mvn clean package`.

This generates a jar file containing the dependencies the Amazon Kinesis Video Streams Edge Agent requires at `kvs-edge-agent/target/libs.jar`.

5. Place the `libs.jar` into the folder that contains the component's artifacts.

   Type `mv ./target/libs.jar ./KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/`.

6. Set environment variables using the values from previous steps. The following table provides descriptions for the variables.

Environment Variable Name	Required	Description
`AWS_REGION`	Yes	The Region that is used. **Example:** us-west-2
`AWS_IOT_CA_CERT`	Yes	File path to the CA certificate used to establish trust with the backend service through TLS. **Example:** `/file/path/to/AmazonRootCA1.pem`
`AWS_IOT_CORE_CERT`	Yes	File path to the X.509 certificate. **Example:** `/file/path/to/certificateID-certificate .pem.crt`
`AWS_IOT_CORE_CREDENTIAL_ENDPOINT`	Yes	The AWS IoT Core credential endpoint provider endpoint for your AWS account. **Example:** `credential-account-specific-prefix .credenti`

Environment Variable Name	Required	Description
		`als.iot.` *`aws-region`* `.amazonaws.com`
`AWS_IOT_CORE_DATA_ATS_ENDPOINT`	Yes	The [AWS IoT Core data plane endpoint](#) for your AWS account. **Example:** *`data-account-specific-prefix`* `.iot.`*`aws-region`* `.amazonaws.com`
`AWS_IOT_CORE_PRIVATE_KEY`	Yes	File path to the private key used in the public/private key pair. For more information, see [Key management in AWS IoT](#). **Example:** `/`*`file`*`/`*`path`*`/`*`to`*`/`*`certificateID-private`* `.pem.key`
`AWS_IOT_CORE_ROLE_ALIAS`	Yes	The name of the role alias pointing to the AWS IAM role to use when connecting to AWS IoT Core. **Example:** `kvs-edge-role-alias`

Environment Variable Name	Required	Description
`AWS_IOT_CORE_THING_NAME`	Yes	The name of the AWS IoT thing that the application is being run on. **Example:** `my-edge-device-thing`
`GST_PLUGIN_PATH`	Yes	File path pointing to the folder that contains the `gstkvssink` and `IngestorPipelineJNI` platform-dependent libraries. Lets GStreamer load these plugins. For more information, see [Download, Build, and Configure the GStreamer Element](). **Example:** /*download-location*/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/*EdgeAgentVersion*/

Environment Variable Name	Required	Description
`LD_LIBRARY_PATH`	Yes	File path pointing to the directory containing the `cproducer` and `KinesisVideoProducer` platform-dependent libraries. **Example:** `/`*`download-location`*`/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/`*`EdgeAgentVersion`*`/lib/`
`AWS_KVS_EDGE_CLOUDWATCH_ENABLED`	No	Determines if the Amazon Kinesis Video Streams Edge Agent will post job health metrics onto Amazon CloudWatch. Accepted values: `TRUE`/`FALSE` (case insensitive). Defaults to `FALSE` if not provided. **Example:** FALSE

Environment Variable Name	Required	Description
AWS_KVS_EDGE_LOG_LEVEL	No	The level of logging the Amazon Kinesis Video Streams Edge Agent outputs. Accepted values: • OFF • ALL • FATAL • ERROR • WARN • INFO, default, if not provided • DEBUG • TRACE **Example:** `INFO`
AWS_KVS_EDGE_LOG_MAX_FILE_SIZE	No	Once the log file reaches this size, a rollover will occur. • **Min:** 0 • **Max:** 10000 • **Default:** 20, if not provided • **Units:** Megabytes (MB) **Example:** 5

Environment Variable Name	Required	Description
AWS_KVS_EDGE_LOG_OUTPUT_DIRECTORY	No	The file path pointing to the directory where the Amazon Kinesis Video Streams Edge Agent logs are output. Defaults to `./log` if not provided. **Example:** `/file/path/`
AWS_KVS_EDGE_LOG_ROLLOVER_COUNT	No	The number of rolled-over logs to keep before deleting. • **Min:** 1 • **Max:** 100 • **Default:** 10, if not provided **Example:** 20
AWS_KVS_EDGE_RECORDING_DIRECTORY	No	File path pointing to the directory recorded media will be written to. Defaults to the current directory if not provided. **Example:** `/file/path/`
GST_DEBUG	No	Specifies the level of GStreamer logs to output. For more information, see the [GStreamer documentation](). **Example:** 0

Environment Variable Name	Required	Description
`GST_DEBUG_FILE`	No	Specifies the output file of the GStreamer debug logs. If unset, debug logs get output to standard error. For more information, see the GStreamer documentation. **Example:** `/tmp/gstreamer-logging .log`

7. Clear the GStreamer cache. Type:

```
rm ~/.cache/gstreamer-1.0/registry.your-os-architecture.bin
```

   For more information, see the GStreamer registry documentation.

8. Prepare and run the java command. The Amazon Kinesis Video Streams Edge Agent accepts the following arguments:

Java Property Name	Required	Description
`java.library.path`	No	File path pointing to the folder containing the `gstkvssink` and `IngestorPipelineJNI` dependent libraries. If not provided, the Amazon Kinesis Video Streams Edge Agent will search for them in the current directory. **Important** The Amazon Kinesis Video Streams

Java Property Name	Required	Description
		Edge Agent won't function correctly if it can't locate these files. **Example:** /*file*/*path*/

To set these, add `-D`*`java-property-name`*`=`*`value`* to the java command used to run the jar.

For example:

```
java -Djava.library.path=/download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion \
  --add-opens java.base/jdk.internal.misc=ALL-UNNAMED \
  -Dio.netty.tryReflectionSetAccessible=true \
  -cp kvs-edge-agent.jar:libs.jar \
  com.amazonaws.kinesisvideo.edge.controller.ControllerApp
```

Important

Run the java command above from the same directory as /*download-location*/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/*EdgeAgentVersion*.

9. Send configurations to the application using the AWS CLI.

   a. Create a new file, *example-edge-configuration*.json.

      Paste the following code into the file. This is a sample configuration that records daily from 9:00:00 AM to 4:59:59 PM (according to the system time on your AWS IoT device). It also uploads the recorded media daily from 7:00:00 PM to 9:59:59 PM.

      For more information, see the section called “StartEdgeConfigurationUpdate”.

```
{
```

```
    "StreamARN": "arn:aws:kinesisvideo:your-region:your-account-id:stream/your-
stream/0123456789012",
    "EdgeConfig": {
        "HubDeviceArn": "arn:aws:iot:your-region:your-account-id:thing/kvs-
edge-agent-demo",
        "RecorderConfig": {
            "MediaSourceConfig": {
                "MediaUriSecretArn": "arn:aws:secretsmanager:your-region:your-
account-id:secret:your-secret-dRbHJQ",
                "MediaUriType": "RTSP_URI"
            },
            "ScheduleConfig": {
                "ScheduleExpression": "0 0 9,10,11,12,13,14,15,16 ? * * *",
                "DurationInSeconds": 3599
            }
        },
        "UploaderConfig": {
            "ScheduleConfig": {
                "ScheduleExpression": "0 0 19,20,21 ? * * *",
                "DurationInSeconds": 3599
            }
        },
        "DeletionConfig": {
            "EdgeRetentionInHours": 15,
            "LocalSizeConfig": {
              "MaxLocalMediaSizeInMB": 2800,
              "StrategyOnFullSize": "DELETE_OLDEST_MEDIA"
            },
            "DeleteAfterUpload": true
        }
    }
}
```

b. To send the file to the Amazon Kinesis Video Streams Edge Agent, type the following in the AWS CLI:

```
aws kinesisvideo start-edge-configuration-update --cli-input-json
 "file://example-edge-configuration.json"
```

10. Repeat the previous step for each stream for the Amazon Kinesis Video Streams Edge Agent.

Step 9: (Optional) Install the CloudWatch agent on the device

> **Note**
>
> Be aware of the CloudWatch quotas.

Follow these procedures to install and configure the CloudWatch agent to automatically upload the logs generated by the Amazon Kinesis Video Streams Edge Agent to CloudWatch.

For procedures to install the CloudWatch agent on your device, see the Amazon CloudWatch User Guide.

When prompted for the configuration, select **one** of the following configurations.

> **Important**
>
> The `file_path` in the following configurations assumes that the default logging output location is used.
> The file path used assumes that you are running the Amazon Kinesis Video Streams Edge Agent from the location: `*download-location*/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/*version*`.

- To configure the CloudWatch agent to upload logs and post device RAM and CPU metrics, paste the following into the configuration file.

```
{
  "agent": {
    "run_as_user": "ubuntu",
    "metrics_collection_interval": 60
  },
  "metrics": {
    "metrics_collected": {
      "mem": {
        "measurement": [
          "mem_used_percent"
        ],
        "append_dimensions": {
          "IotThing": "YourIotThingName"
        }
```

```
      },
      "cpu": {
        "resources": [
          "*"
        ],
        "measurement": [
          "usage_active"
        ],
        "totalcpu": true,
        "append_dimensions": {
          "IotThing": "YourIotThingName"
        }
      }
    }
  },
  "logs": {
    "logs_collected": {
      "files": {
        "collect_list": [
          {
            "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/java_kvs.log",
            "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
            "log_stream_name": "YourIotThingName-java_kvs.log"
          },
          {
            "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvs_edge.log*",
            "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
            "log_stream_name": "YourIotThingName-cpp_kvs_edge.log"
          },
          {
            "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvs_streams.log*",
            "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
            "log_stream_name": "YourIotThingName-cpp_kvs_streams.log"
          },
          {
            "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvssink.log*",
            "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
            "log_stream_name": "YourIotThingName-cpp_kvssink.log"
          }
        ]
```

```
      }
    }
  }
}
```

- To upload only the logs and not collect device's RAM and CPU, use the following configuration:

```
{
  "logs": {
    "logs_collected": {
      "files": {
        "collect_list": [
          {
            "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/java_kvs.log",
            "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
            "log_stream_name": "YourIotThingName-java_kvs.log"
          },
          {
            "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvs_edge.log*",
            "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
            "log_stream_name": "YourIotThingName-cpp_kvs_edge.log"
          },
          {
            "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvs_streams.log*",
            "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
            "log_stream_name": "YourIotThingName-cpp_kvs_streams.log"
          },
          {
            "file_path": "download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/version/log/cpp_kvssink.log*",
            "log_group_name": "/aws/kinesisvideo/EdgeRuntimeAgent",
            "log_stream_name": "YourIotThingName-cpp_kvssink.log"
          }
        ]
      }
    }
  }
}
```

Step 10: (Optional) Run the Amazon Kinesis Video Streams Edge Agent as a native process

Set up the Amazon Kinesis Video Streams Edge Agent as a systemd service.

`systemd` is a systems and service manager on Linux devices. `systemd` is the recommended way to manage the process, as it will restart the Amazon Kinesis Video Streams Edge Agent in case the application encounters an error or the device running the application loses power.

Do the following:

Run the Amazon Kinesis Video Streams Edge Agent as a native process

1. Create a new file in `/etc/systemd/system` and name it `aws.kinesisvideo.edge-runtime-agent.service`.

   Paste the following:

```
[Unit]
Description=AWS Kinesis Video Streams edge agent
After=network.target
StartLimitBurst=3
StartLimitInterval=30

[Service]
Type=simple
Restart=on-failure
RestartSec=10
WorkingDirectory=/download-location/kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion
Environment="GST_PLUGIN_PATH=/download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion"
Environment="LD_LIBRARY_PATH=/download-location/kvs-edge-agent/KvsEdgeComponent/
artifacts/aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib"
...
Environment="AWS_IOT_CORE_DATA_ATS_ENDPOINT=data-account-specific-prefix.iot.aws-
region.amazonaws.com"
ExecStart=/usr/lib/jvm/java-11-amazon-corretto/bin/java --add-opens java.base/
jdk.internal.misc=ALL-UNNAMED -Dio.netty.tryReflectionSetAccessible=true -cp kvs-
edge-agent.jar:libs.jar com.amazonaws.kinesisvideo.edge.controller.ControllerApp

[Install]
```

```
WantedBy=multi-user.target
```

For more information about the parameters accepted by `systemd` service configuration file, see the documentation.

> **Note**
>
> Add required the environment variables at the `...` location, as specified in the section called "8. Build and run the Amazon Kinesis Video Streams Edge Agent".

2. Reload the service files to include the new service.

   Type `sudo systemctl daemon-reload`.

3. Start the service.

   Type `sudo systemctl start aws.kinesisvideo.edge-runtime-agent.service`.

4. Check the status of the Amazon Kinesis Video Streams Edge Agent service to verify that it is running.

   Type `sudo systemctl status aws.kinesisvideo.edge-runtime-agent.service`.

   The following is an example of the output that you will see.

```
aws.kinesisvideo.edge-runtime-agent.service - AWS Kinesis Video Streams edge agent
     Loaded: loaded (/etc/systemd/system/aws.kinesisvideo.edge-runtime-
agent.service; disabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-06-08 19:15:02 UTC; 6s ago
   Main PID: 506483 (java)
      Tasks: 23 (limit: 9518)
     Memory: 77.5M
        CPU: 4.214s
     CGroup: /system.slice/aws.kinesisvideo.edge-runtime-agent.service
             ##506483 /usr/lib/jvm/java-11-amazon-corretto/bin/java -cp kvs-edge-
agent.jar:libs.jar com.amazonaws.kinesisvideo.edge.controller.ControllerApp
```

5. Inspect the logs for any errors.

   Type `journalctl -e -u aws.kinesisvideo.edge-runtime-agent.service`.

6. Type `systemctl --help` for the full list of options to manage the process using `systemctl`.

The following are some common commands to manage the Amazon Kinesis Video Streams Edge Agent:

- To restart, type `sudo systemctl restart` *`aws.kinesisvideo.edge-runtime-agent`*`.service`.
- To stop, type `sudo systemctl stop` *`aws.kinesisvideo.edge-runtime-agent`*`.service`.
- To automatically start on every device reboot, type `sudo systemctl enable` *`aws.kinesisvideo.edge-runtime-agent`*`.service`.

Deploy the Amazon Kinesis Video Streams Edge Agent to AWS IoT Greengrass

Follow these steps to deploy the Amazon Kinesis Video Streams Edge Agent to AWS IoT Greengrass to record and upload media from IP cameras.

Topics

Step 1: Create an Ubuntu Amazon EC2 instance

Do the following to create an Ubuntu Amazon EC2 instance.

Create an Ubuntu Amazon EC2 instance

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

   Verify that the appropriate Region is selected.

2. Choose **Launch Instance**.

   Complete the following fields:

   - **Name –** Type a name for the instance.
   - **Application and OS Images (Amazon Machine Image)** – Select **Ubuntu**.
   - **Instance type** – Select **t2.large**.
   - **Key pair login –** Create your own key pair.
   - **Network settings** – Keep the default.
   - **Configure storage** – Increase the volume to 256 GiB.
   - **Advanced settings** – Keep the default.

3. Launch the instance and SSH into it.

   Do the following:

   1. Select **Instances** in the left navigation, then select the instance ID.
   2. Choose **Connect** in the top-right.
   3. Choose **SSH client** and follow the instructions on the screen.
   4. Open a terminal and navigate to the downloaded `.pem` file (likely in `~/Downloads`).
   5. The first time you follow these procedures, you will receive the message "The authenticity of host (...) can't be established." Type **yes**.

4. Install system libraries to build the Amazon Kinesis Video Streams Edge Agent onto the instance.

```
wget -O- https://apt.corretto.aws/corretto.key | sudo apt-key add -
sudo add-apt-repository 'deb https://apt.corretto.aws stable main'

sudo apt-get update

sudo apt-get install -y gcc libssl-dev libcurl4-openssl-dev liblog4cplus-dev \
libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev \
```

```
gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-bad \
gstreamer1.0-plugins-good gstreamer1.0-tools \
unzip java-11-amazon-corretto-jdk maven
```

> **Important**
>
> If you see a screen telling you that some services need to be restarted, press Enter to select **Ok**.

For more information, see *Amazon Corretto 11 User Guide*.

Step 2: Set up the AWS IoT Greengrass V2 core device on the device

Follow these procedures to install the AWS IoT Greengrass core nucleus software on the Amazon EC2 instance.

Set up the AWS IoT Greengrass core device

1. Sign in to the AWS Management Console, https://console.aws.amazon.com/iot/.

   Verify that the appropriate Region is selected.
2. In the left navigation, select **Greengrass devices**, **Core devices**.
3. Choose **Set up one core device**.
4. Complete the steps on the screen.

   - **Step 1: Register a Greengrass core device**. Type a name for the device.
   - **Step 2: Add to a thing group to apply a continuous deployment**. Select **No group**.
   - **Step 3: Install the Greengrass Core software**. Select **Linux**.
     - **Step 3.1: Install Java on the device**

       Java is installed as part of the section called "1. Create Ubuntu instance". Return to that step if you don't have Java installed yet.
     - **Step 3.2: Copy AWS credentials onto the device**

       Open the `bash/zsh` option and paste the export commands in the Amazon EC2 instance.
     - **Step 3.3: Run the installer**

1. Copy and run the **Download the installer** and **Run the installer** commands in the Ubuntu Amazon EC2 instance.

   > **Note**
   >
   > The **Run the installer** command will automatically update based on the name you chose in a previous step.

2. Make note of the token exchange service (TES) role that is created. You need it later.

> **Note**
>
> By default, the role created is called **GreengrassV2TokenExchangeRole**.

Step 3: Create the Amazon Kinesis Video Streams and AWS Secrets Manager resources for your IP camera RTSP URLs

Follow these procedures to create the streams and secrets needed in AWS Secrets Manager. Do this step first, because you need the ARNs of the created resources in the policies.

Create Amazon Kinesis Video Streams

Create Amazon Kinesis Video Streams using the AWS Management Console, AWS CLI, or API.

In the AWS Management Console, open the Amazon Kinesis Video Streams console. Choose **Video streams** in the left navigation.

For more information, see the section called “Create a Kinesis video stream”.

Create secrets in AWS Secrets Manager

In the AWS Management Console, open the AWS Secrets Manager console. Choose **Secrets** in the left navigation.

Verify that the appropriate Region is selected.

1. Choose **Store a new secret**.

   a. **Step 1: Choose secret type**

- Select **Other type of secret**.
- In the **Key/Value Pairs** section, add a key-value pair.

  Key: `MediaURI`

  > **Note**
  >
  > The key must be `MediaURI`. This is case-sensitive. If you enter it incorrectly, the application doesn't work.

  Value: *`Your MediaURI`*.

  Example

  Example: `rtsp://<YourCameraIPAddress>:<YourCameraRTSPPort>/YourCameraMediaURI`.

b. **Step 2: Configure secret**. Give this secret a name. Name it whatever you want.

c. **Step 3: Configure rotation - optional**. Choose **Next**.

d. **Step 4: Review**. Choose **Store**.

2. If your secret does not display immediately, select the refresh button.

   Choose the name of your secret. Make note of the **Secret ARN**.

3. Repeat this process for each MediaURI that you want to stream from.

> **Note**
>
> The AWS network blocks some public RTSP sources. You cannot access these from within the Amazon EC2 instance or if you are running unmanaged while connected to the VPN.
>
> > **Important**
> >
> > Your camera RTSP URL should stream video in h.264 format. The fragment duration must not exceed the limit mentioned in the section called "Producer SDK limits".
> > Amazon Kinesis Video Streams Edge Agent only supports video.

Run `gst-discoverer-1.0 Your RtspUrl` to make sure that your camera is reachable from your device.

Save the ARNs for all of the streams and secrets that you created. You need these for the next step.

Step 4: Add permissions to the token exchange service (TES) role

Grant the token exchange service (TES) role to the device that assumes permissions to look at the secrets. This is necessary for the AWS Secrets Manager AWS IoT Greengrass component to work correctly.

Add permissions to the TES role

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Roles** in the left navigation and search for the TES role that you created earlier in the process.
3. In the **Add permissions** dropdown, select **Attach policies**.
4. Choose **Create policy**.
5. Scroll down and select **Edit**.
6. In the policy editor, choose **JSON** and edit the policy.

   Replace the policy with the following:

   > **Note**
   >
   > Replace `arn:aws:kinesisvideo:*:*:stream/streamName1/*` and `arn:aws:kinesisvideo:*:*:stream/streamName2/*` with the ARNs for the streams that you created in a previous step.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:ListStreams"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:DescribeStream",
                "kinesisvideo:PutMedia",
                "kinesisvideo:TagStream",
                "kinesisvideo:GetDataEndpoint"
            ],
            "Resource": [
                "arn:aws:kinesisvideo:*:*:stream/streamName1/*",
                "arn:aws:kinesisvideo:*:*:stream/streamName2/*"
            ]
        }
    ]
}
```

7. On the **Add tags** page, choose **Next: Review**.
8. Name your policy, then choose **Create policy**.

   An example of a policy name is **KvsEdgeAccessPolicy**.
9. Close the tab and return to the tab where you were attaching a policy to the TES role.

   Choose the refresh button, then search for the newly created policy.

   Select the check box and choose **Attach policies**.

   On the next screen, you see a note that says **Policy was successfully attached to role.**
10. Create and attach another policy, this time for your secrets.

    Replace the policy with the following:

> **Note**
>
> Replace `arn:aws:secretsmanager:*:*:secret:*` with the ARNs containing the MediaURI secrets that you created in the section called "3. Create resources for your IP camera RTSP URLs".

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:*:*:secret:*",
                "arn:aws:secretsmanager:*:*:secret:*"
            ]
        }
    ]
}
```

11. Create and attach another policy, this time for Amazon CloudWatch metrics. Replace the policy with the following:

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

Step 5: Install the AWS IoT Greengrass Secret Manager component on the device

The Amazon Kinesis Video Streams Edge Agent requires the AWS IoT Greengrass Secret Manager component to be installed on the device first.

Install the Secret Manager component

1. Sign in to the AWS Management Console and open the AWS IoT Core console at https://console.aws.amazon.com/iot/. Verify that the appropriate Region is selected.
2. In the left navigation, choose **Greengrass devices**, **Deployments**.

   Choose the deployment with the same target as the thing we created in the section called “2. Set up the AWS IoT Greengrass core device”.
3. In the **Actions** dropdown in the top right corner, choose **Revise**.

   In the pop-up that appears, choose **Revise deployment**.
4. Complete the following sections:
   - **Step 1: Specify target**. Choose **Next**.
   - **Step 2: Select components**.
     - Verify that the **aws.greengrass.Cli** component is selected. Do not uninstall this component.
     - Toggle the **Show only selected components** switch and search for **aws.greengrass.SecretManager**.
     - Check the box next to **aws.greengrass.SecretManager**, then choose **Next**.
   - **Step 3: Configure components**. Configure the AWS IoT Greengrass Secret Manager component to download the secrets from within the AWS IoT Greengrass environment.

     Select the **aws.greengrass.SecretManager** component, then choose **Configure component**.

     In the screen that appears, update the AWS Secrets Manager ARNs in the **Configuration to merge** box.

> **Note**
>
> Replace `arn:aws:secretsmanager:*:*:secret:*` with the ARNs of the secrets that you created in the section called "3. Create resources for your IP camera RTSP URLs".

```
{
 "cloudSecrets": [
      {
        "arn": "arn:aws:secretsmanager:*:*:secret:*"
      },
      {
        "arn": "arn:aws:secretsmanager:*:*:secret:*"
      }
    ]
}
```

> **Note**
>
> `cloudSecrets` is a list of objects with the key `arn`. For more information, see the Secret manager configuration section in the AWS IoT Greengrass Version 2 Developer Guide.

When you're done, select **Confirm**, then choose **Next**.

- **Step 4: Configure advanced settings**. Select **Next**.
- **Step 5: Review**. Select **Deploy**.

5. Confirm that the AWS Secrets Manager component and permissions were installed correctly.

   On the Ubuntu Amazon EC2 instance, type `sudo /greengrass/v2/bin/greengrass-cli component details --name aws.greengrass.SecretManager` to verify that the component received the updated configuration.

6. Inspect the AWS IoT Greengrass core logs.

   Type `sudo less /greengrass/v2/logs/greengrass.log`.

Review for deployment errors.

If there was an error, revise the deployment to remove the `aws.greengrass.SecretManager` component.

Type `sudo service greengrass restart` to restart the AWS IoT Greengrass core service.

If the deployment error was related to missing permissions, review the the section called "4. Add permissions to the TES role" section to make sure that the TES role has the proper permissions. Then, repeat this section.

Update the secrets on the AWS IoT Greengrass Secret Manager component

> **Important**
>
> The AWS IoT Greengrass Secret Manager component fetches and caches secrets only when the deployment is updated.

In order to update the secrets on the AWS IoT Greengrass Secret Manager component, follow the preceding steps 1–6, with the following change.

Step 3: Configure components. Configure the AWS IoT Greengrass Secret Manager component to download the secrets from within the AWS IoT Greengrass environment.

Select the **aws.greengrass.SecretManager** component, then choose **Configure component**.

In the screen that appears, paste `[""]` in the **Reset paths** box, and update the AWS Secrets Manager ARNs in the **Configuration to merge** box.

For more information, see Reset updates.

Step 6: Deploy the Amazon Kinesis Video Streams Edge Agent AWS IoT Greengrass component on the device

Deploy the Amazon Kinesis Video Streams Edge Agent AWS IoT Greengrass component on the device

1. Download the `tar` file using the provided link.

   If you completed the Amazon Kinesis Video Streams Edge Agent interest form, check your email for the download link. If you haven't completed the form, complete it here.
2. Verify the checksum.
3. Extract the binaries and jar in your device.

   Type: `tar -xvf kvs-edge-agent.tar.gz`.

   After extraction, your folder structure will look like the following:

```
kvs-edge-agent/LICENSE
kvs-edge-agent/THIRD-PARTY-LICENSES
kvs-edge-agent/pom.xml
kvs-edge-agent/KvsEdgeComponent
kvs-edge-agent/KvsEdgeComponent/recipes
kvs-edge-agent/KvsEdgeComponent/recipes/recipe.yaml
kvs-edge-agent/KvsEdgeComponent/artifacts
kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/edge_log_config

kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/kvs-edge-agent.jar
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/libgstkvssink.so
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/libIngestorPipelineJNI.so
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib/libcproducer.so
```

```
kvs-edge-agent/KvsEdgeComponent/artifacts/
aws.kinesisvideo.KvsEdgeComponent/EdgeAgentVersion/lib/libKinesisVideoProducer.so
```

> **Note**
>
> The release folder name should be set up in a way that reflects the latest binary release number. For example, a 1.0.0 release will have the folder name set as 1.0.0.

4. Build the dependencies jar.

   > **Note**
   >
   > The jar included with the kvs-edge-agent.tar.gz does not have the dependencies. Use the following steps to build those libraries.

   Navigate to the `kvs-edge-agent` folder that contains `pom.xml`.

   Type `mvn clean package`.

   This will generate a jar file containing the dependencies the Amazon Kinesis Video Streams Edge Agent requires at `kvs-edge-agent/target/libs.jar`.

5. Place the libs.jar into the folder that contains the component's artifacts.

   Type `mv ./target/libs.jar ./KvsEdgeComponent/artifacts/ aws.kinesisvideo.KvsEdgeComponent/`*`EdgeAgentVersion`*`/`.

6. **Optional.** Configure properties. The Amazon Kinesis Video Streams Edge Agent accepts the following environment variables in AWS IoT Greengrass mode:

Environment Variable Name	Required	Description
`AWS_REGION`	Yes	The Region that is used. **Example:** us-west-2 AWS IoT Greengrass Core software automatically

Environment Variable Name	Required	Description
		sets this value for you. For more information, see the Component environment variable reference topic in the AWS IoT Greengrass Version 2 Developer Guide.
`GST_PLUGIN_PATH`	Yes	File path pointing to the folder containing the `gstkvssink` and `IngestorPipelineJNI` platform-dependent libraries. This lets GStreamer load these plugins. For more information, see Download, Build, and Configure the GStreamer Element. **Example:** `/`*`download-location`*`/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/`*`EdgeAgentVersion`*`/`

Environment Variable Name	Required	Description
`LD_LIBRARY_PATH`	Yes	File path pointing to the directory containing the `cproducer` and `KinesisVideoProducer` platform-dependent libraries. **Example:** `/`*`download-location`*`/kvs-edge-agent/KvsEdgeComponent/artifacts/aws.kinesisvideo.KvsEdgeComponent/`*`EdgeAgentVersion`*`/lib/`
`AWS_KVS_EDGE_CLOUDWATCH_ENABLED`	No	Determines if the Amazon Kinesis Video Streams Edge Agent will post job health metrics onto Amazon CloudWatch. Accepted values: `TRUE`/`FALSE` (case insensitive). Defaults to `FALSE` if not provided. **Example:** FALSE

Environment Variable Name	Required	Description
AWS_KVS_EDGE_LOG_LEVEL	No	The level of logging the Amazon Kinesis Video Streams Edge Agent outputs. **Accepted values:** • OFF • ALL • FATAL • ERROR • WARN • INFO, default, if not provided • DEBUG • TRACE **Example:** INFO
AWS_KVS_EDGE_LOG_MAX_FILE_SIZE	No	Once the log file reaches this size, a rollover will occur. • **Min:** 1 • **Max:** 100 • **Default:** 20, if not provided • **Units:** Megabytes (MB) **Example:** 5

Environment Variable Name	Required	Description
AWS_KVS_EDGE_LOG_OUTPUT_DIRECTORY	No	The file path pointing to the directory where the Amazon Kinesis Video Streams Edge Agent logs are output. Defaults to `./log` if not provided. **Example:** `/file/path/`
AWS_KVS_EDGE_LOG_ROLLOVER_COUNT	No	The number of rolled-over logs to keep before deleting. • **Min:** 1 • **Max:** 100 • **Default:** 10, if not provided **Example:** 20
AWS_KVS_EDGE_RECORDING_DIRECTORY	No	File path pointing to the directory recorded media will be written to. Defaults to the current directory if not provided. **Example:** `/file/path/`
GREENGRASS_ROOT_DIRECTORY	No	The file path to the AWS IoT Greengrass root directory. This defaults to `/greengrass/v2/` if not provided. **Example:** `/file/path/`

Environment Variable Name	Required	Description
GST_DEBUG	No	Specifies the level of GStreamer logs to output. For more information, see the [GStreamer documentation](). **Example:** 0
GST_DEBUG_FILE	No	Specifies the output file of the GStreamer debug logs. If unset, debug logs get output to standard error. For more information, see the [GStreamer documentation](). **Example:** `/`*`tmp`*`/`*`gstreamer-logging`*` .log`

Open `kvs-edge-agent/KvsEdgeComponent/recipes/recipe.yaml` and modify the run script to add any of the preceding environment variables.

> **Important**
>
> Make sure that the modified run script doesn't contain any **tab** characters. The AWS IoT Greengrass core software won't be able to read the recipe.

7. Deploy the Amazon Kinesis Video Streams Edge Agent AWS IoT Greengrass component.

   Type:

```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
  --recipeDir <download location>/kvs-edge-agent/KvsEdgeComponent/recipes/ \
  --artifactDir <download location>/kvs-edge-agent/KvsEdgeComponent/artifacts/ \
  --merge "aws.kinesisvideo.KvsEdgeComponent=EdgeAgentVersion"
```

For additional information, see the following sections in the *AWS IoT Greengrass Version 2 Developer Guide*:

- AWS IoT Greengrass CLI commands
- Deploy AWS IoT Greengrass components to devices

8. Send configurations to the application using the AWS CLI.

   a. Create a new file, *example-edge-configuration*.json.

      Paste the following code into the file. This is a sample configuration that records daily from 9:00:00 AM to 4:59:59 PM (according to the system time on your AWS IoT device). It also uploads the recorded media daily from 7:00:00 PM to 9:59:59 PM.

      For more information, see the section called "StartEdgeConfigurationUpdate".

```
{
    "StreamARN": "arn:aws:kinesisvideo:your-region:your-account-id:stream/your-stream/0123456789012",
    "EdgeConfig": {
        "HubDeviceArn": "arn:aws:iot:your-region:your-account-id:thing/kvs-edge-agent-demo",
        "RecorderConfig": {
            "MediaSourceConfig": {
                "MediaUriSecretArn": "arn:aws:secretsmanager:your-region:your-account-id:secret:your-secret-dRbHJQ",
                "MediaUriType": "RTSP_URI"
            },
            "ScheduleConfig": {
                "ScheduleExpression": "0 0 9,10,11,12,13,14,15,16 ? * * *",
                "DurationInSeconds": 3599
            }
        },
        "UploaderConfig": {
            "ScheduleConfig": {
                "ScheduleExpression": "0 0 19,20,21 ? * * *",
                "DurationInSeconds": 3599
            }
        },
        "DeletionConfig": {
            "EdgeRetentionInHours": 15,
            "LocalSizeConfig": {
```

```
                "MaxLocalMediaSizeInMB": 2800,
                "StrategyOnFullSize": "DELETE_OLDEST_MEDIA"
              },
              "DeleteAfterUpload": true
          }
      }
}
```

b. Type the following in the AWS CLI to send the file to the Amazon Kinesis Video Streams Edge Agent:

```
aws kinesisvideo start-edge-configuration-update --cli-input-json
 "file://example-edge-configuration.json"
```

9. Repeat the previous step for each stream for the Amazon Kinesis Video Streams Edge Agent.

Step 7: (Optional) Install the AWS IoT Greengrass log manager component on the device

> **Note**
>
> Be aware of the CloudWatch quotas.

Follow these procedures to configure the Amazon Kinesis Video Streams Edge Agent logs to automatically upload to CloudWatch using the AWS IoT Greengrass log manager component.

Install the AWS IoT Greengrass log manager component

1. Confirm that the AWS IoT Greengrass device role has the appropriate permissions.

   a. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

   b. Click **Roles** in the left navigation.

   c. Choose the name of the TES role created in the section called "2. Set up the AWS IoT Greengrass core device". Use the search bar if necessary.

   d. Select the `GreengrassV2TokenExchangeRoleAccess` policy.

   e. Select the JSON tab and verify that the policy looks like the following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams",
                "s3:GetBucketLocation"
            ],
            "Resource": "*"
        }
    ]
}
```

   f. If the `GreengrassV2TokenExchangeRoleAccess` policy doesn't exist, or if some required permissions are missing, create a new IAM policy with these permissions and attach it to the TES role created in the section called “2. Set up the AWS IoT Greengrass core device”.

2. Sign in to the AWS Management Console and open the AWS IoT Core console at https://console.aws.amazon.com/iot/. Verify that the appropriate Region is selected.

3. In the left navigation, choose **Greengrass devices**, **Deployments**.

   Choose the deployment with the same target as the thing you created in the section called “2. Set up the AWS IoT Greengrass core device”.

4. In the top right corner, select **Actions**, then choose **Revise**.

   In the pop-up that appears, choose **Revise deployment**.

5. Complete the following sections:

   a. **Step 1: Specify target.** Choose **Next**.

   b. **Step 2: Select components.**

      i. Verify that the **aws.greengrass.Cli** component and **aws.greengrass.SecretManager** components are still selected.

> **Important**
>
> Don't uninstall these components.

ii. Toggle the **Show only selected components** switch and search for **aws.greengrass.LogManager**.

iii. Select the box next to **aws.greengrass.LogManager**, then choose **Next**.

c. **Step 3: Configure components.** Configure the AWS IoT Greengrass log manager component to upload the logs generated by the Amazon Kinesis Video Streams Edge Agent.

Select the **aws.greengrass.LogManager** component, then choose **Configure component**.

In the screen that appears, paste the following log manager configuration in the **Configuration to merge** box.

```
{
    "logsUploaderConfiguration": {
        "componentLogsConfigurationMap": {
            "aws.kinesisvideo.KvsEdgeComponent/java_kvs.log": {
                "diskSpaceLimit": "100",
                "diskSpaceLimitUnit": "MB",
                "logFileDirectoryPath": "/greengrass/v2/work/
aws.kinesisvideo.KvsEdgeComponent/log",
                "logFileRegex": "java_kvs.log\\w*"
            },
            "aws.kinesisvideo.KvsEdgeComponent/cpp_kvs_edge.log": {
                "diskSpaceLimit": "100",
                "diskSpaceLimitUnit": "MB",
                "logFileDirectoryPath": "/greengrass/v2/work/
aws.kinesisvideo.KvsEdgeComponent/log",
                "logFileRegex": "cpp_kvs_edge.log\\w*"
            },
            "aws.kinesisvideo.KvsEdgeComponent/cpp_kvssink.log": {
                "diskSpaceLimit": "100",
                "diskSpaceLimitUnit": "MB",
                "logFileDirectoryPath": "/greengrass/v2/work/
aws.kinesisvideo.KvsEdgeComponent/log",
                "logFileRegex": "cpp_kvssink.log\\w*"
            },
```

```
            "aws.kinesisvideo.KvsEdgeComponent/cpp_kvs_streams.log": {
                "diskSpaceLimit": "100",
                "diskSpaceLimitUnit": "MB",
                "logFileDirectoryPath": "/greengrass/v2/work/
aws.kinesisvideo.KvsEdgeComponent/log",
                "logFileRegex": "cpp_kvs_streams.log\\w*"
            }
        }
    },
    "periodicUploadIntervalSec": "1"
}
```

Important

The `logFileDirectoryPath` in the preceding configuration assumes that the default logging output location is used.

Note

For more information about each of the parameters for the log manager configuration, see the Log manager section of the AWS IoT Greengrass Version 2 Developer Guide.

Once you finish, select **Confirm**, then choose **Next**.

d. **Step 4: Configure advanced settings.** Select **Next**.

e. **Step 5: Review.** Select **Deploy**.

6. Confirm that the AWS log manager component and permissions were installed correctly.
7. On the Ubuntu Amazon EC2 instance, type `sudo /greengrass/v2/bin/greengrass-cli component details --name aws.greengrass.LogManager` to verify the component received the updated configuration.
8. Inspect the AWS IoT Greengrass core logs.

   Type `sudo less /greengrass/v2/logs/greengrass.log`.

   Review for deployment errors.

If there was an error, revise the deployment to remove the `aws.greengrass.LogManager` component.

Type `sudo service greengrass restart` to restart the AWS IoT Greengrass core service.

If the deployment error was related to missing permissions, review the section called "4. Add permissions to the TES role" to make sure that the TES role has proper permissions. Then, repeat this section.

Amazon Kinesis Video Streams Edge Agent FAQ

The following are some common questions for the Amazon Kinesis Video Streams Edge Agent service.

What operating systems does Amazon Kinesis Video Streams Edge Agent support?

Amazon Kinesis Video Streams Edge Agent currently supports the following operating systems:

Ubuntu

- 22.x
  - AMD64
- 18.x
  - ARM

AL2

- amzn2
  - AMD64 amazonlinux:2.0.20210219.0-amd64 (Snowball)

Does the Amazon Kinesis Video Streams Edge Agent support H.265 media?

Amazon Kinesis Video Streams Edge Agent only supports H.264 elementary streams.

Does the Amazon Kinesis Video Streams Edge Agent work in AL2?

Yes.

How can I run multiple streams within the AWS IoT thing or device?

Send another the section called "StartEdgeConfigurationUpdate" to the same `HubDeviceArn`, but different Amazon Kinesis Video Streams/AWS Secrets Manager ARNs.

How can I edit a `StartEdgeConfigurationUpdate` after it has been sent?

Send an updated the section called "StartEdgeConfigurationUpdate"to the same `HubDeviceArn` with the same Amazon Kinesis Video Streams ARN. When the application receives the message from Amazon Kinesis Video Streams, it overrides the previous configuration for that stream. Changes will take place then.

Do you have any examples of common `ScheduleConfigs`?

The Amazon Kinesis Video Streams Edge Agent uses the system time of the device that it's running on.

Description	ScheduleExpression	DurationInSeconds
24/7 recording, hourly uploading	`(null ScheduleConfig)`	
9:00:00 AM - 4:59:59 PM every day	`0 0 9-16 * * ? *`	3599
9:00:00 AM - 4:59:59 PM weekdays	`0 0 9-16 ? * 2-6 *`	3599
	`0 0 9-16 ? * 2,3,4,5,6 *`	3599
	`0 0 9-16 ? * MON-FRI *`	3599
	`0 0 9-16 ? * MON,TUE,WED,THU,FRI *`	3599

Description	ScheduleExpression	DurationInSeconds
9:00:00 AM - 4:59:59 PM weekends	`0 0 9-16 ? * SAT,SUN *`	3599
10:00:00 PM - 11:59:59 PM weekdays	`0 0 22,23 ? * MON-FRI *`	3599
9:00:00 AM - 10:00:00 AM every day	`0 0 9 * * ? *`	3600
4:00:00 PM - 5:59:59 PM every day	`0 0 16-17 * * ? *`	3599

For more examples, see the Quartz documentation.

Is there a maximum stream limit?

The Amazon Kinesis Video Streams Edge Agent currently has a hard limit of 16 streams per device. Use the the section called "DeleteEdgeConfiguration" API to delete streams from a device. Updating a configuration for the same stream using the the section called "StartEdgeConfigurationUpdate" does not increase the device's stream count.

How do I restart a job that has errored out?

If an error is encountered, the Amazon Kinesis Video Streams Edge Agent will attempt to restart the job. However, with some errors (such as configuration errors), you must manually restart the job.

To determine which jobs need to be restarted manually, see the **FatalError** metric in the section called "Monitoring the Amazon Kinesis Video Streams Edge Agent with CloudWatch".

Resend the the section called "StartEdgeConfigurationUpdate" to restart the job for the stream.

How do I monitor the health of my Amazon Kinesis Video Streams Edge Agent?

For more information, see the section called "Monitoring the Amazon Kinesis Video Streams Edge Agent with CloudWatch".

Stream video through a VPC

This beta is available in preview in the Europe (Paris) Region, eu-west-3. To access these components and our getting started guide, email us.

Amazon Kinesis Video Streams VPC endpoint service allows you to stream and consume video through the Amazon network without any data going through the public internet.

To request access, email us the following information:

- Account ID
- Stream ARNs
- VPC ID

> **Note**
>
> It may take up to a week for us to add you to the service.

If you haven't worked with VPC endpoints in the past, review the following information to get familiar with the concept:

- AWS PrivateLink background
- VPC getting started guide

Additional information

Once you've been added to the beta, we will email you a link to additional information about this feature.

VPC endpoint procedures

Quotas

The primary quota differences are:

- Lower quota for all bandwidth APIs (2 mbps):
  - PutMedia
  - GetMedia
  - GetMediaForFragmentList
- 10 streams allowed per customer

Create an endpoint

Once you're allow listed, you will receive the VPC endpoint service name for Amazon Kinesis Video Streams. It will look like `com.amazonaws.region.kinesisvideo`.

Create an interface VPC endpoint for Amazon Kinesis Video Streams using either the Amazon VPC Console or the AWS Command Line Interface (AWS CLI).

In the AWS CLI, type the following:

```
aws ec2 create-vpc-endpoint \
--vpc-id customer-provided-vpc-id\
--service-name com.amazonaws.eu-west-2.kinesisvideo \
--private-dns-enabled
```

> **Important**
>
> Traffic within your VPC will use private DNS to route over the endpoint. If you don't enable this, you'll need to implement your own DNS logic. For more information about private DNS, see AWS PrivateLink documentation.

For more information on the AWS CLI option, see create-vpc-endpoint.

Control access to endpoints

You can attach an endpoint policy to your VPC endpoint that controls access to Amazon Kinesis Video Streams. The policy specifies the following information:

- the principal that can perform actions,
- the actions that can be performed, and
- the resources on which actions can be performed.

For more information, see Controlling access to services with VPC endpoints using endpoint policies in the AWS PrivateLink Guide.

The following is an example of an endpoint policy for Amazon Kinesis Video Streams. When attached to an endpoint, this policy denies access to the listed `PutMedia` actions for all principals on all resources.

```
{
"Statement":[
      {
         "Principal":"*",
         "Effect":"Deny",
         "Action":[
            "kinesisvideo:PutMedia"
         ],
         "Resource":"*"
      }
   ]
}
```

Images in Kinesis Video Streams

You can use Amazon Kinesis Video Streams APIs and SDKs to help you extract images from your video streams. You can use these images for enhanced playback applications such as thumbnails or enhanced scrubbing, or for use in machine learning pipelines. Kinesis Video Streams offers on-demand image extraction through an API, or automated image extraction from metadata tags in an ingested video.

For information about using Kinesis Video Streams managed support for images, see:

- **On-demand Image Generation (GetImages)** - This API allows customers to extract a single image or multiple images from video stored in Kinesis Video Streams.
- **Automated Image Generation (S3 Delivery)** - Configure Kinesis Video Streams to automatically extract images from video data in real time based on tags in an uploaded video, and deliver the images to the customer-specified S3 bucket.

Topics

Getting started with GetImages

Managed support for Images provides a fully managed way to get images from the video data streamed and stored in Kinesis Video Streams. You can use images to run machine learning (ML) workloads such as person, pet, or vehicle detection. Images can also be used to add interactive elements to playback, such as image previews for motion events and scrubbing for a video clip.

For more information about the `GetImages` feature, see GetImages in the *Amazon Kinesis Video Streams Archived Media* API reference guide.

Getting started with Amazon S3 delivery

Currently, customers run and manage their own image transcoding pipeline to create images for various purposes like scrubbing, image preview, running ML models on images, and more. Kinesis Video Streams offers the capability to transcode and deliver the images. Kinesis Video Streams will automatically extract images from video data in real-time based on a tag, and deliver the images to a customer specified S3 bucket.

UpdateImageGenerationConfiguration

To set up a Kinesis video stream to enable image generation to Amazon S3:

1. Create an **S3 Bucket** for image generation based on the tags added in the SDK using the new API. Note the **S3 URI**, which is required in the next step, when updating the image generation configurations for the streams.
2. Create a JSON file called **update-image-generation-input.json** with the following content as input.

```
{
 "StreamName": "TestStream",
 "ImageGenerationConfiguration":
 {
  "Status": "ENABLED",
  "DestinationConfig":
  {
   "DestinationRegion": "us-east-1",
   "Uri": "s3://bucket-name"
  },
  "SamplingInterval": 200,
  "ImageSelectorType": "PRODUCER_TIMESTAMP",
  "Format": "JPEG",
  "FormatConfig": {
                "JPEGQuality": "80"
       },
  "WidthPixels": 320,
  "HeightPixels": 240
 }
}
```

You can use the AWS CLI to invoke the `UpdateImageGenerationConfiguration` API operation to add the Amazon S3 ARN created previously and change the status to ENABLED.

```
aws kinesisvideo update-image-generation-configuration \
--cli-input-json file://./update-image-generation-input.json \
```

Request:

```
UpdateImageGenerationConfiguration HTTP/1.1

  Method: 'POST'
  Path: '/updateImageGenerationConfiguration'
  Body: {
    StreamName: 'String', // Optional. Either stream name or arn should be passed
    StreamArn: 'String', // Optional. Either stream name or arn should be passed
    ImageGenerationConfiguration : {
        // required
        Status: 'Enum', // ENABLED | DISABLED,
       ImageSelectorType: 'Enum', // SERVER_TIMESTAMP | PRODUCER_TIMESTAMP..
        DestinationConfig: {
            DestinationRegion: 'String',
            Uri: string,
        },
        SamplingInterval: 'Number'//
        Format: 'Enum', // JPEG | PNG
        // Optional parameters
        FormatConfig: {
                'String': 'String',
       },
        WidthPixels: 'Number', // 1 - 3840 (4k).
        HeightPixels: 'Number' // 1 - 2160 (4k).
    }
  }
```

Response:

```

  HTTP/1.1 200
  Content-type: application/json
  Body: {
  }
```

> **Note**
>
> It takes at least 1 minute to initiate the image generation workflow after updating the image generation configuration. Wait at least 1 minute before invoking `PutMedia` after the update call.

DescribeImageGenerationConfiguration

To view image generation configurations that are already set for a stream, customers can make a `DescribeImageGenerationConfiguration` request, as follows.

Request:

```
DescribeImageGenerationConfiguration HTTP/1.1


 Method: 'POST'
 Path: '/describeImageGenerationConfiguration'
 Body: {
   StreamName: 'String', // Optional. Either stream name or arn should be passed
   StreamArn: 'String', // Optional. Either stream name or arn should be passed
 }
```

Response:

```
HTTP/1.1 200
  Content-type: application/json
  Body: {
    ImageGenerationConfiguration : {
        Status: 'Enum',
        ImageSelectorType: 'Enum', // SERVER_TIMESTAMP | PRODUCER_TIMESTAMP
        DestinationConfig: {
            DestinationRegion: 'String'
            Uri: 'string',
        },
        SamplingInterval: 'Number',
        Format: 'Enum',
```

```
        FormatConfig: {
                'String': 'String',
        },
        WidthPixels: 'Number',
        HeightPixels: 'Number'
    }
  }
```

To learn more about the `DescribeImageGenerationConfiguration` feature, see [DescribeImageGenerationConfiguration](#) in the *Amazon Kinesis Video Streams Developer Guide.*

Producer MKV tags

You can use the Kinesis Video Streams Producer SDK to tag specific fragments of interest by exposing an API operation in the SDK. For an example of a tag, see [this code](#). Upon calling this API, the SDK will add a set of predefined MKV tags along with the fragment data. Kinesis Video Streams will recognize these special MKV tags and initiate the image generation workflow based on the image processing configuration of that stream.

Any fragment metadata provided along with the Amazon S3 image generation tags will be saved as Amazon S3 metadata.

Syntax for Producer MKV tags

```
|+ Tags
| + Tag
|  // MANDATORY: Predefined MKV tag to trigger image generation for the fragment
|  + Simple
|   + Name: AWS_KINESISVIDEO_IMAGE_GENERATION

|  // OPTIONAL: S3 prefix which will be set as prefix for generated image.
|  + Simple
|   + Name: AWS_KINESISVIDEO_IMAGE_PREFIX
|   + String: image_prefix_in_s3 // 256 bytes max m

| // OPTIONAL: Key value pairs that will be persisted as S3 Image object metadata.
|  + Simple
|   + Name: CUSTOM_KEY_1 // Max 128 bytes
|   + String:CUSTOM_VALUE_1 // Max 256 bytes
|  + Simple
```

```
|    + Name: CUSTOM_KEY_2 // Max 128 bytes
|    + String: CUSTOM_VALUE_2 // Max 256 bytes
```

Adding metadata tags in Producer SDK using PutEventMetaData

The `PutEventMetaData` function appends an MKV file that's associated with an event. `PutEventMetaData` takes two parameters. The first parameter is an event whose value comes from the `STREAM_EVENT_TYPE` enum. The second parameter, [`pStreamEventMetadata`](), is optional and can be used to include additional metadata as key-value pairs. There is a limit of five key-value pairs of metadata that can be added.

Limits

The following table lists the limitations associated with the metadata tags. If the metadata tag limit is adjustable, you can request an increase through your account manager.

Limit	Max value	Adjustable
Image prefix length	256	no
Optional metadata key length	128	no
Optional metadata value length	256	no
Max number of optional metadata	10	yes

S3 Object metadata

By default, Kinesis Video Streams will set the **fragment number**, **producer**, and **server timestamp** of the image generated as Amazon S3 object metadata. If any additional fragment data is specified in the MKV tags, those tags will also be added to the Amazon S3 object metadata. The following example shows the correct syntax for the Amazon S3 object metadata.

```
{
```

```
    // KVS S3 object metadata
    x-amz-meta-aws_kinesisvideo_fragment_number : 'string',
    x-amz-meta-aws_kinesisvideo_producer_timestamp: 'number',
    x-amz-meta-aws_kinesisvideo_server_timestamp: 'number',

    // Optional key value pair sent as part of the MKV tags
    custom_key_1: custom_value_1,
    custom_key_2: custom_value_2,
}
```

S3 object path (image)

The following list shows the correct format of the object path and describes each element within the path.

Format:

ImagePrefix_AccountID_StreamName_ImageTimecode_RandomID.file-extension

1.`ImagePrefix` - Value of `AWS_KINESISVIDEO_IMAGE_PREFIX`.

2.`AccountID` - Account ID under which the stream is created.

3.`StreamName` - Name of the stream for which the image is generated.

4.`ImageTimecode` - Epoch timecode in the fragment at which the image is generated.

5.`RandomID` - Random GUID.

6.`file-extension` - JPG or PNG based on the image format requested.

Amazon S3 URI recommendations to protect against throttling

If you write thousands of images to Amazon S3, there's a risk of throttling. For more information, see S3 Prefix Put Request Limits.

An Amazon S3 prefix starts with a PUT limit of 3,500 PUT requests per second, and will gradually ramp up over time for unique prefixes. Avoid using dates and times as Amazon S3 prefixes. Time coded data will impact one prefix at a time, and will also change regularly, invalidating previous prefix scale ups. To enable faster, consistent Amazon S3 scaling, we recommend adding a random prefix, like a hex code or UUID to the Amazon S3 Destination URI. For example, hex code prefixes will naturally split your requests randomly among 16 different prefixes (a prefix for each unique hex character), which will allow a 56,000 PUT requests per second after Amazon S3 has auto-scaled.

Notifications in Kinesis Video Streams

When a media fragment is available for consumption, Kinesis Video Streams notifies customers using Amazon Simple Notification Service (Amazon SNS) notifications. The following topic explains how to get started with notifications.

UpdateNotificationConfiguration

Use this API operation to update the notification information for a stream. For more information about the `UpdateNotificationConfiguration` feature, see UpdateNotificationConfiguration in the *Amazon Kinesis Video Streams Developer Guide*.

> **Note**
>
> It takes at least one minute to initiate the notification after updating the notification configuration. Wait at least one minute before invoking `PutMedia` after the update call.

DescribeNotificationConfiguration

Use this API to describe a notification configuration attached to a stream. For more information about the `DescribeNotificationConfiguration` feature, see DescribeNotificationConfiguration in the *Amazon Kinesis Video Streams Developer Guide*.

Producer MKV tags

You can use the Kinesis Video Streams Producer SDK to tag specific fragments of interest by exposing an API operation in the SDK. See a sample of how this works in this section of code. Upon calling this API, the SDK will add a set of predefined MKV tags along with the fragment data. Kinesis Video Streams will recognize these special MKV tags and initiate notifications for the tagged fragments.

Any fragment metadata provided along with the Notification MKV tags will be published as part of the Amazon SNS topic payload.

Syntax for Producer MKV tags

```
|+ Tags
| + Tag
|  // MANDATORY: Predefined MKV tag to trigger the notification for the fragment
|  + Simple
|   + Name: AWS_KINESISVIDEO_NOTIFICATION
|   + String
| // OPTIONAL: Key value pairs that will be sent as part of the Notification payload
|  + Simple
|   + Name: CUSTOM_KEY_1 // Max 128 bytes
|   + String:CUSTOM_VALUE_1 // Max 256 bytes
|  + Simple
|   + Name: CUSTOM_KEY_2 // Max 128 bytes
|   + String: CUSTOM_VALUE_2 // Max 256 bytes
```

MKV tag limits

The following table lists the limitations associated with the metadata tags. If the metadata tag limit is adjustable, you can request an increase through your account manager.

Limit	Max value	Adjustable
Optional metadata key length	128	No
Optional metadata value length	256	No
Max number of optional metadata	10	Yes

Amazon SNS topic payload

Any notification initiated through the previous workflow will deliver the Amazon SNS topic payload, as shown in the following example. This example is an Amazon SNS message that occurs after consuming notification data from an Amazon Simple Queue Service (Amazon SQS) queue.

```
{
"Type" : "Notification",
"MessageId" : Message ID,
"TopicArn" : SNS ARN,
"Subject" : "Kinesis Video Streams Notification",
"Message" : "{\"StreamArn\":\Stream Arn,\"FragmentNumber\":\Fragment Number,
\"FragmentStartProducerTimestamp\":FragmentStartProducerTimestamp,
                \"FragmentStartServerTimestamp\":FragmentStartServerTimestamp,
\"NotificationType\":\"PERSISTED\",\"NotificationPayload\":{\ CUSTOM_KEY_1:
\CUSTOM_VALUE_1,
                \CUSTOM_KEY_2:\CUSTOM_VALUE_2}}",
"Timestamp" : "2022-04-25T18:36:29.194Z",
"SignatureVersion" : Signature Version,
"Signature" : Signature,
"SigningCertURL" : Signing Cert URL,
"UnsubscribeURL" : Unsubscribe URL
}
```

```
Subject: "Kinesis Video Streams Notification"
Message:
{
    "StreamArn":Stream Arn,
    "FragmentNumber":Fragment Number,
    "FragmentStartProducerTimestamp":Fragment Start Producer Timestamp,
    "FragmentStartServerTimestamp":Fragment Start Server Timestamp,
    "NotificationType":"PERSISTED",
    "NotificationPayload":{
        CUSTOM_KEY_1:CUSTOM_VALUE_1,
        CUSTOM_KEY_2:CUSTOM_VALUE_2
    }
}
```

Viewing your Amazon SNS messages

You cannot read messages directly from an Amazon SNS topic because there's no API for doing so. To view the messages, subscribe an SQS queue to the SNS topic, or choose any other Amazon SNS supported destination. However, the most efficient option for viewing messages is to use Amazon SQS.

To view your Amazon SNS messages using Amazon SQS

1. Create an Amazon SQS queue.
2. From the AWS Management Console, open the Amazon SNS topic set as a destination under `NotificationConfiguration`.
3. Choose **Create Subscription**, and then choose the Amazon SQS queue created in the first step.
4. Run a `PutMedia` session with the Notification configuration enabled and with the Notification MKV tags added to the fragments.
5. Choose the Amazon SQS queue in the Amazon SQS console, and then select **Send and receive messages** for the Amazon SQS queue.
6. Poll for messages. This command should show all notifications generated by the `PutMedia` session. For information about polling, see Amazon SQS short and long polling.

Security in Amazon Kinesis Video Streams

Cloud security at AWS is the highest priority. As an AWS customer, you will benefit from a data center and network architecture 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. The effectiveness of our security is regularly tested and verified by third-party auditors as part of the AWS compliance programs. To learn about the compliance programs that apply to Kinesis Video Streams, 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 organization's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Kinesis Video Streams. The following topics show you how to configure Kinesis Video Streams to meet your security and compliance objectives. You'll also learn how to use other AWS services that can help you to monitor and secure your Kinesis Video Streams resources.

Topics

Data protection in Kinesis Video Streams

You can use server-side encryption (SSE) using AWS Key Management Service (AWS KMS) keys to meet strict data management requirements by encrypting your data at rest in Amazon Kinesis Video Streams.

Topics

What is server-side encryption for Kinesis Video Streams?

Server-side encryption is a feature in Kinesis Video Streams that automatically encrypts data before it's stored at rest using an AWS KMS key that you specify. Data is encrypted before it's written to the Kinesis Video Streams stream storage layer, and it's decrypted after it's retrieved from storage. As a result, your data is always encrypted at rest within the Kinesis Video Streams service.

With server-side encryption, your Kinesis video stream producers and consumers don't need to manage KMS keys or cryptographic operations. If data retention is enabled, your data is automatically encrypted as it enters and leaves Kinesis Video Streams, so your data at rest is encrypted. AWS KMS provides all the keys that are used by the server-side encryption feature. AWS KMS streamlines the use of a KMS key for Kinesis Video Streams that's managed by AWS, a user-specified AWS KMS key imported into the AWS KMS service.

Costs, Regions, and performance considerations

When you apply server-side encryption, you are subject to AWS KMS API usage and key costs. Unlike custom AWS KMS keys, the `(Default) aws/kinesis-video` KMS key is offered with no charge. However, you still must pay for the API usage costs that Kinesis Video Streams incurs on your behalf.

API usage costs apply for every KMS key, including custom ones. The AWS KMS costs scale with the number of user credentials that you use on your data producers and consumers because each user credential requires a unique API call to AWS KMS.

The following describes the costs by resource:

Keys

- The KMS key for Kinesis Video Streams that's managed by AWS (alias = `aws/kinesis-video`) has no charge.
- User-generated KMS keys are subject to AWS KMS key costs. For more information, see AWS Key Management Service Pricing.

AWS KMS API usage

API requests to generate new data encryption keys or to retrieve existing encryption keys increase as traffic increases, and are subject to AWS KMS usage costs. For more information, see AWS Key Management Service Pricing: Usage.

Kinesis Video Streams generates key requests even when retention is set to 0 (no retention).

Availability of server-side encryption by Region

Server-side encryption of Kinesis video streams is available in all the AWS Regions where Kinesis Video Streams is available.

How do I get started with server-side encryption?

Server-side encryption is always enabled on Kinesis Video Streams. If a user-provided key isn't specified when the stream is created, the AWS managed key (provided by Kinesis Video Streams) is used.

A user-provided KMS key must be assigned to a Kinesis video stream when it's created. You can't assign a different key to a stream using the UpdateStream API later on.

You can assign a user-provided KMS key to a Kinesis video stream in two ways:

- When creating a Kinesis video stream in the AWS Management Console, specify the KMS key in the **Encryption** tab on the **Create a new video stream** page.

- When creating a Kinesis video stream using the [CreateStream] API, specify the key ID in the `KmsKeyId` parameter.

Creating and using a customer managed key

This section describes how to create and use your own KMS keys instead of using the key administered by Amazon Kinesis Video Streams.

Creating a customer managed key

For information about how to create your own keys, see Creating Keys in the *AWS Key Management Service Developer Guide*. After you create keys for your account, the Kinesis Video Streams service returns these keys in the **Customer managed keys** list.

Using a customer managed key

After the correct permissions are applied to your consumers, producers, and administrators, you can use custom KMS keys in your own AWS account or another AWS account. All KMS keys in your account appear in the **Customer managed keys** list on the console.

To use custom KMS keys that are located in another account, you must have permissions to use those keys. You must also create the stream using the `CreateStream` API. You can't use KMS keys from different accounts in streams created in the console.

> **Note**
>
> The KMS key isn't accessed until the `PutMedia` or `GetMedia` operation is carried out. This has the following results:
>
> - If the key that you specify doesn't exist, the `CreateStream` operation succeeds, but `PutMedia` and `GetMedia` operations on the stream fail.
> - If you use the provided key (`aws/kinesis-video`), the key isn't present in your account until the first `PutMedia` or `GetMedia` operation is performed.

Permissions to use a customer managed key

Before you can use server-side encryption with a customer managed key, you must configure KMS key policies to allow encryption of streams and encryption and decryption of stream records. For examples and more information about AWS KMS permissions, see AWS KMS API Permissions: Actions and Resources Reference.

> **Note**
>
> The use of the default service key for encryption doesn't require application of custom IAM permissions.

Before you use a customer managed key, verify that your Kinesis video stream producers and consumers (IAM principals) are users in the AWS KMS default key policy. Otherwise, writes and reads from a stream will fail, which could ultimately result in data loss, delayed processing, or hung applications. You can manage permissions for KMS keys using IAM policies. For more information, see Using IAM Policies with AWS KMS.

Example producer permissions

Your Kinesis video stream producers must have the `kms:GenerateDataKey` permission:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey"
        ],
        "Resource": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    {
        "Effect": "Allow",
        "Action": [
            "kinesis-video:PutMedia",
        ],
        "Resource": "arn:aws:kinesis-video:*:123456789012:MyStream"
    }
  ]
}
```

Example consumer Permissions

Your Kinesis video stream consumers must have the `kms:Decrypt` permission:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt"
        ],
        "Resource": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    {
        "Effect": "Allow",
        "Action": [
            "kinesis-video:GetMedia",
        ],
        "Resource": "arn:aws:kinesis-video:*:123456789012:MyStream"
    }
  ]
}
```

Controlling access to Kinesis Video Streams resources using IAM

You can use AWS Identity and Access Management (IAM) with Amazon Kinesis Video Streams, to control whether users in your organization can perform a task using specific Kinesis Video Streams API operations and whether they can use specific AWS resources.

For more information about IAM, see the following:

- AWS Identity and Access Management (IAM)
- Getting started
- IAM User Guide

Contents

Policy syntax

An IAM policy is a JSON document that consists of one or more statements. Each statement is structured as follows:

```
{
  "Statement":[{
    "Effect":"effect",
    "Action":"action",
    "Resource":"arn",
    "Condition":{
      "condition":{
        "key":"value"
        }
      }
    }
  ]
}
```

There are various elements that make up a statement:

- **Effect** – The *effect* can be `Allow` or `Deny`. By default, users don't have permission to use resources and API actions, so all requests are denied. An explicit allow overrides the default. An explicit deny overrides any allows.
- **Action** – The *action* is the specific API action for which you are granting or denying permission.
- **Resource** – The resource that's affected by the action. To specify a resource in the statement, you must use its Amazon Resource Name (ARN).
- **Condition** – Conditions are optional. They can be used to control when your policy is in effect.

As you create and manage IAM policies, we recommend that you use the IAM Policy Generator and the IAM Policy Simulator.

Actions for Kinesis Video Streams

In an IAM policy statement, you can specify any API action from any service that supports IAM. For Kinesis Video Streams, use the following prefix with the name of the API action: `kinesisvideo:`. For example: `kinesisvideo:CreateStream`, `kinesisvideo:ListStreams`, and `kinesisvideo:DescribeStream`.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": ["kinesisvideo:action1", "kinesisvideo:action2"]
```

You can also specify multiple actions using wildcards. For example, you can specify all actions whose name begins with the word "Get" as follows:

```
"Action": "kinesisvideo:Get*"
```

To specify all Kinesis Video Streams operations, use the asterisk (*) wildcard as follows:

```
"Action": "kinesisvideo:*"
```

For the complete list of Kinesis Video Streams API actions, see the *Kinesis Video Streams API reference*.

Amazon Resource Names (ARNs) for Kinesis Video Streams

Each IAM policy statement applies to the resources that you specify using their ARNs.

Use the following ARN resource format for Kinesis Video Streams:

```
arn:aws:kinesisvideo:region:account-id:stream/stream-name/code
```

For example:

```
"Resource": arn:aws:kinesisvideo:*:111122223333:stream/my-stream/0123456789012
```

You can get the ARN of a stream using DescribeStream.

Granting other IAM accounts access to a Kinesis video stream

You might need to grant permission to other IAM accounts to perform operations on streams in Kinesis Video Streams. The following overview describes the general steps to grant access to video streams across accounts:

1. Get the 12-digit account ID of the account that you want to grant permissions to perform operations on the stream resource created in your account.

   Example: In the following steps, we'll use 111111111111 as the account ID for the account that you want to grant permission to, and 999999999999 as the ID for your Kinesis Video Streams

2. Create an IAM managed policy in the account that owns the stream (999999999999) that allows the level of access that you want to grant.

   Sample policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:GetDataEndpoint",
                "kinesisvideo:DescribeStream",
                "kinesisvideo:PutMedia"
            ],
            "Resource": "arn:aws:kinesisvideo:us-west-2:999999999999:stream/custom-stream-name/1613732218179"
        }
    ]
}
```

   For other example policies for Kinesis Video Streams resources, see Example Policies in the next section.

3. Create a role in the account that owns the stream (999999999999), and specify the account that you want to grant permissions for (111111111111). This will add a trusted entity to the role.

   Sample trusted policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Attach the policy that you created in the previous step to this role.

You've now created a role in account 999999999999 which has the permission to operations like `DescribeStream`, `GetDataEndpoint`, and `PutMedia` on a stream resource ARN in the managed policy. This new role also trusts the other account, 111111111111, to assume this role.

> **Important**
>
> Make note of the role ARN, you'll need it in the next step.

4. Create a managed policy in the other account, 111111111111, that allows the `AssumeRole` action on the role that you created in account 999999999999 in the previous step. You'll need to mention the role ARN from the previous step.

   Sample policy:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::999999999999:role/CustomRoleName"
    }
}
```

5. Attach the policy created in the previous step to an IAM entity, like a role or a user in account 111111111111. This user now has the permission to assume role `CustomRoleName` in account 999999999999.

   This user's crendentials call AWS STS `AssumeRole` API to get the session credentials, which are subsequently used to call Kinesis Video Streams APIs on the stream created in account 999999999999.

```
aws sts assume-role --role-arn "arn:aws:iam::999999999999:role/CustomRoleName" --
role-session-name "kvs-cross-account-assume-role"
{
    "Credentials": {
        "AccessKeyId": "",
        "SecretAccessKey": "",
        "SessionToken": "",
        "Expiration": ""
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "",
        "Arn": ""
    }
}
```

6. Set the access key, secret key, and session credentials based on the previous set in the environment.

```
set AWS_ACCESS_KEY_ID=
set AWS_SECRET_ACCESS_KEY=
set AWS_SESSION_TOKEN=
```

7. Run Kinesis Video Streams APIs to describe and get the data endpoint for the stream in account 999999999999.

```
aws kinesisvideo describe-stream --stream-arn "arn:aws:kinesisvideo:us-
west-2:999999999999:stream/custom-stream-name/1613732218179"
{
    "StreamInfo": {
        "StreamName": "custom-stream-name",
        "StreamARN": "arn:aws:kinesisvideo:us-west-2:999999999999:stream/custom-
stream-name/1613732218179",
        "KmsKeyId": "arn:aws:kms:us-west-2:999999999999:alias/aws/kinesisvideo",
        "Version": "abcd",
```

```
        "Status": "ACTIVE",
        "CreationTime": "2018-02-19T10:56:58.179000+00:00",
        "DataRetentionInHours": 24
    }
}

aws kinesisvideo get-data-endpoint --stream-arn "arn:aws:kinesisvideo:us-
west-2:999999999999:stream/custom-stream-name/1613732218179" --api-name "PUT_MEDIA"
{
    "DataEndpoint": "https://s-b12345.kinesisvideo.us-west-2.amazonaws.com"
}
```

For generic step-by-step instructions on granting cross-account access, see Delegate Access Across AWS accounts Using IAM Roles.

Example policies for Kinesis Video Streams

The following example policies demonstrate how you can control user access to your Kinesis Video Streams

Example 1: Allow users to get data from any Kinesis video stream

This policy allows a user or group to perform the `DescribeStream`, `GetDataEndpoint`, `GetMedia`, `ListStreams`, and `ListTagsForStream` operations on any Kinesis video stream. This policy is appropriate for users who can get data from any video stream.

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

Example 2: Allow a user to create a Kinesis video stream and write data to it

This policy allows a user or group to perform the `CreateStream` and `PutMedia` operations. This policy is appropriate for a security camera that can create a video stream and send data to it.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:CreateStream",
                "kinesisvideo:PutMedia"
            ],
            "Resource": "*"
        }
    ]
}
```

Example 3: Allow a user full access to all Kinesis Video Streams resources

This policy allows a user or group to perform any Kinesis Video Streams operation on any resource. This policy is appropriate for administrators.

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

Example 4: Allow a user to write data to a specific Kinesis video stream

This policy allows a user or group to write data to a specific video stream. This policy is appropriate for a device that can send data to a single stream.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "kinesisvideo:PutMedia",
            "Resource": "arn:aws:kinesisvideo:us-west-2:123456789012:stream/
your_stream/0123456789012"
        }
    ]
}
```

Controlling access to Kinesis Video Streams resources using AWS IoT

This section describes how to enable a device (for example, a camera) to send audio and video data to one particular Kinesis video stream only. You can do this by using the AWS IoT credentials provider and an AWS Identity and Access Management (IAM) role.

Devices can use X.509 certificates to connect to AWS IoT using TLS mutual authentication protocols. Other AWS services (for example, Kinesis Video Streams) don't support certificate-based authentication, but can be called using AWS credentials in AWS Signature Version 4 format. The Signature Version 4 algorithm typically requires the caller to have an access key ID and a secret access key. AWS IoT has a credentials provider that allows you to use the built-in X.509 certificate as the unique device identity to authenticate AWS requests (for example, requests to Kinesis Video Streams). This removes the need to store an access key ID and a secret access key on your device.

The credentials provider authenticates a client (in this case, a Kinesis Video Streams SDK that's running on the camera that you want to send data to a video stream) using an X.509 certificate and issues a temporary, limited-privilege security token. You can use the token to sign and authenticate any AWS request (in this case, a call to the Kinesis Video Streams). For more information, see Authorizing Direct Calls to AWS Services.

This way of authenticating your camera's requests to Kinesis Video Streams requires you to create and configure an IAM role and attach appropriate IAM policies to the role so that the AWS IoT credentials provider can assume the role on your behalf.

For more information about AWS IoT, see AWS IoT Core Documentation. For more information about IAM, see AWS Identity and Access Management (IAM).

Topics

AWS IoT ThingName as stream name

Topics

Step 1: Create an AWS IoT thing type and an AWS IoT thing

In AWS IoT, a thing is a representation of a specific device or logical entity. In this case, an AWS IoT thing represents your Kinesis video stream that you want to configure resource-level access control. In order to create a thing, first, you must create an AWS IoT thing type. You can use AWS IoT thing types to store description and configuration information that's common to all things associated with the same thing type.

1. The following example command creates a thing type `kvs_example_camera`:

   ```
   aws --profile default iot create-thing-type --thing-type-name kvs_example_camera >
    iot-thing-type.json
   ```

2. This example command creates the `kvs_example_camera_stream` thing of the `kvs_example_camera` thing type:

   ```
   aws --profile default  iot create-thing --thing-name kvs_example_camera_stream --
   thing-type-name kvs_example_camera > iot-thing.json
   ```

Step 2: Create an IAM role to be assumed by AWS IoT

IAM roles are similar to users, in that a role is an AWS identity with permissions policies that determine what the identity can and can't do in AWS. A role can be assumed by anyone who needs it. When you assume a role, it provides you with temporary security credentials for your role session.

The role that you create in this step can be assumed by AWS IoT to obtain temporary credentials from the security token service (STS) when performing credential authorization requests from the client. In this case, the client is the Kinesis Video Streams SDK that's running on your camera.

Perform the following steps to create and configure this IAM role:

1. Create an IAM role.

   The following example command creates an IAM role called `KVSCameraCertificateBasedIAMRole`:

   ```
   aws --profile default iam create-role --role-name KVSCameraCertificateBasedIAMRole
    --assume-role-policy-document 'file://iam-policy-document.json' > iam-role.json
   ```

   You can use the following trust policy JSON for the `iam-policy-document.json`:

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "credentials.iot.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

2. Next, attach a permissions policy to the IAM role that you previously created. This permissions policy allows selective access control (a subset of supported operations) for an AWS resource. In this case, the AWS resource is the video stream that you want your camera to send data. In other words, once all the configuration steps are complete, this camera will be able to send data only to this video stream.

   ```
   aws --profile default iam put-role-policy --role-name
    KVSCameraCertificateBasedIAMRole --policy-name KVSCameraIAMPolicy --policy-
   document 'file://iam-permission-document.json'
   ```

You can use the following IAM policy JSON for the iam-permission-document.json:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:DescribeStream",
                "kinesisvideo:PutMedia",
                "kinesisvideo:TagStream",
                "kinesisvideo:GetDataEndpoint"
            ],
            "Resource": "arn:aws:kinesisvideo:*:*:stream/${credentials-
iot:ThingName}/*"
        }
    ]
}
```

Note that this policy authorizes the specified actions only on a video stream (AWS resource) that is specified by the placeholder (${credentials-iot:ThingName}). This placeholder takes on the value of the AWS IoT thing attribute `ThingName` when the AWS IoT credentials provider sends the video stream name in the request.

3. Next, create a Role Alias for your IAM role. Role alias is an alternate data model that points to the IAM role. An AWS IoT credentials provider request must include a role-alias to indicate which IAM role to assume to obtain the temporary credentials from the STS.

   The following sample command creates a role alias called `KvsCameraIoTRoleAlias`,

   ```
   aws --profile default iot create-role-alias --role-alias KvsCameraIoTRoleAlias --
   role-arn $(jq --raw-output '.Role.Arn' iam-role.json) --credential-duration-seconds
    3600 > iot-role-alias.json
   ```

4. Now you can create the policy that will enable AWS IoT to assume role with the certificate (once it is attached) using the role alias.

   The following sample command creates a policy for AWS IoT called `KvsCameraIoTPolicy`.

```
aws --profile default iot create-policy --policy-name KvsCameraIoTPolicy --policy-
document 'file://iot-policy-document.json'
```

You can use the following command to create the iot-policy-document.json document JSON:

```
cat > iot-policy-document.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:AssumeRoleWithCertificate"
            ],
            "Resource": "$(jq --raw-output '.roleAliasArn' iot-role-alias.json)"
        }
    ]
}
EOF
```

Step 3: Create and configure the X.509 certificate

Communication between a device (your video stream) and AWS IoT is protected through the use of X.509 certificates.

1. Create the certificate to which you must attach the policy for AWS IoT that you previously created.

```
aws --profile default iot create-keys-and-certificate --set-as-active --
certificate-pem-outfile certificate.pem --public-key-outfile public.pem.key --
private-key-outfile private.pem.key > certificate
```

2. Attach the policy for AWS IoT (`KvsCameraIoTPolicy` created previously) to this certificate.

```
aws --profile default iot attach-policy --policy-name KvsCameraIoTPolicy --target
 $(jq --raw-output '.certificateArn' certificate)
```

3. Attach your AWS IoT thing (`kvs_example_camera_stream`) to the certificate you just created:

```
aws --profile default iot attach-thing-principal --thing-name
 kvs_example_camera_stream --principal $(jq --raw-output '.certificateArn'
 certificate)
```

4. To authorize requests through the AWS IoT credentials provider, you need the AWS IoT credentials endpoint, which is unique to your AWS account ID. You can use the following command to get the AWS IoT credentials endpoint.

```
aws --profile default iot describe-endpoint --endpoint-type iot:CredentialProvider
 --output text > iot-credential-provider.txt
```

5. In addition to the X.509 certificate created previously, you must also have a CA certificate to establish trust with the backend service through TLS. You can get the CA certificate using the following command:

```
curl --silent 'https://www.amazontrust.com/repository/SFSRootCAG2.pem' --output
 cacert.pem
```

Step 4: Test the AWS IoT credentials with your Kinesis video stream

Now you can test the AWS IoT credentials that you've set up so far.

1. First, create a Kinesis video stream that you want to test this configuration with.

> **Important**
>
> Create a video stream with a name that is identical to the AWS IoT thing name that you created in the previous step (`kvs_example_camera_stream`).

```
aws kinesisvideo create-stream --data-retention-in-hours 24 --stream-name
 kvs_example_camera_stream
```

2. Next, call the AWS IoT credentials provider to get the temporary credentials:

```
curl --silent -H "x-amzn-iot-thingname:kvs_example_camera_stream" --cert
 certificate.pem --key private.pem.key https://IOT_GET_CREDENTIAL_ENDPOINT/role-
aliases/KvsCameraIoTRoleAlias/credentials --cacert ./cacert.pem > token.json
```

Note

You can use the following command to get the IOT_GET_CREDENTIAL_ENDPOINT:

```
IOT_GET_CREDENTIAL_ENDPOINT=`cat iot-credential-provider.txt`
```

The output JSON contains the accessKey, secretKey, and the sessionToken, which you can use to access the Kinesis Video Streams.

3. For your test, you can use these credentials to invoke the Kinesis Video Streams DescribeStream API for the sample `kvs_example_camera_stream` video stream.

```
AWS_ACCESS_KEY_ID=$(jq --raw-output '.credentials.accessKeyId' token.json)
 AWS_SECRET_ACCESS_KEY=$(jq --raw-output '.credentials.secretAccessKey' token.json)
 AWS_SESSION_TOKEN=$(jq --raw-output '.credentials.sessionToken' token.json) aws
 kinesisvideo describe-stream --stream-name kvs_example_camera_stream
```

Step 5: Deploying AWS IoT certificates and credentials on your camera's file system and streaming data to your video stream

Note

The steps in this section describe sending media to a Kinesis video stream from a camera that's using the the section called "C++ Producer Library".

1. Copy the X.509 certificate, the private key, and the CA certificate generated in the previous steps to your camera's file system. Specify the paths for where these files are stored, the role alias name, and the AWS IoT credentials endpoint for running the `gst-launch-1.0` command or your sample application.
2. The following sample command uses AWS IoT certificate authorization to send video to Kinesis Video Streams:

```
gst-launch-1.0 rtspsrc location=rtsp://YourCameraRtspUrl short-header=TRUE !
 rtph264depay ! video/x-h264,format=avc,alignment=au ! h264parse ! kvssink stream-
name="kvs_example_camera_stream" aws-region="YourAWSRegion" iot-certificate="iot-
```

```
certificate,endpoint=credential-account-specific-prefix.credentials.iot.aws-
region.amazonaws.com,cert-path=/path/to/certificate.pem,key-path=/path/to/
private.pem.key,ca-path=/path/to/cacert.pem,role-aliases=KvsCameraIoTRoleAlias"
```

AWS IoT CertificateId as stream name

To represent your device (for example, your camera) through an AWS IoT thing, but authorize a different stream name, then you can use the AWS IoT `certificateId` attribute as your stream name and provide Kinesis Video Streams permissions on the stream using AWS IoT. The steps for accomplishing this are similar to the ones previously outlined, with a few changes.

- Modify the permissions policy to your IAM role (iam-permission-document.json) as follows:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:DescribeStream",
                "kinesisvideo:PutMedia",
                "kinesisvideo:TagStream",
                "kinesisvideo:GetDataEndpoint"
            ],
            "Resource": "arn:aws:kinesisvideo:*:*:stream/${credentials-
iot:AwsCertificateId}/*"
        }
    ]
}
```

Note

The resource ARN uses certificate ID as the placeholder for the stream name. The IAM permission will work when you use the certificate ID as the stream name. Get the certificate ID from the certificate so that you can use that as stream name in the following describe stream API call.

```
export CERTIFICATE_ID=`cat certificate | jq --raw-output '.certificateId'`
```

- Verify this change using the Kinesis Video Streams describe-stream CLI command:

```
AWS_ACCESS_KEY_ID=$(jq --raw-output '.credentials.accessKeyId' token.json)
 AWS_SECRET_ACCESS_KEY=$(jq --raw-output '.credentials.secretAccessKey' token.json)
 AWS_SESSION_TOKEN=$(jq --raw-output '.credentials.sessionToken' token.json) aws
 kinesisvideo describe-stream --stream-name ${CERTIFICATE_ID}
```

- Pass the certificateId to the AWS IoT credentials provider in the sample application in the Kinesis Video Streams C++ SDK:

```
credential_provider =
 make_unique<IotCertCredentialProvider>(iot_get_credential_endpoint,
        cert_path,
        private_key_path,
        role_alias,
        ca_cert_path,
        certificateId);
```

> **Note**
>
> Note that you're passing the thingname to the AWS IoT credentials provider. You can use getenv to pass the thingname to the demo application similar to passing the other AWS IoT attributes. Use the certificate ID as the stream name in the command line parameters when you are running the sample application.

Use AWS IoT credentials to stream to a hard-coded stream name

To represent your device (for example, your camera) through an AWS IoT thing, but authorize streaming to a specific Amazon Kinesis video stream, provide Amazon Kinesis Video Streams permissions on the stream using AWS IoT. The process is similar to the previous sections, with a few changes.

Modify the permissions policy to your IAM role (`iam-permission-document.json`) as follows:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "kinesisvideo:DescribeStream",
                "kinesisvideo:PutMedia",
                "kinesisvideo:TagStream",
                "kinesisvideo:GetDataEndpoint"
            ],
            "Resource": "arn:aws:kinesisvideo:*:*:stream/YourStreamName/*"
        }
    ]
}
```

Copy the X.509 certificate, private key, and CA certificate generated in the previous steps to your camera's file system.

Specify the paths for where these files are stored, the role alias name, the AWS IoT thing name, and the AWS IoT credentials endpoint for running the `gst-launch-1.0` command or your sample application.

The following sample command uses AWS IoT certificate authorization to send video to Amazon Kinesis Video Streams:

```
gst-launch-1.0 rtspsrc location=rtsp://YourCameraRtspUrl short-header=TRUE !
 rtph264depay ! video/x-h264,format=avc,alignment=au ! h264parse ! kvssink
 stream-name="YourStreamName" aws-region="YourAWSRegion" iot-certificate="iot-
certificate,endpoint=credential-account-specific-prefix.credentials.iot.aws-
region.amazonaws.com,cert-path=/path/to/certificate.pem,key-path=/path/to/
private.pem.key,ca-path=/path/to/cacert.pem,role-aliases=KvsCameraIoTRoleAlias,iot-
thing-name=YourThingName"
```

Monitoring Amazon Kinesis Video Streams

Kinesis Video Streams provides monitoring functionality for your delivery streams. For more information, see *Monitoring*.

Compliance Validation for Amazon Kinesis Video Streams

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.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.

- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in Amazon Kinesis Video Streams

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 Kinesis Video Streams

As a managed service, Amazon Kinesis Video Streams is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access Kinesis Video Streams through the network. Clients must support Transport Layer Security (TLS) 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). 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's associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Security best practices for Kinesis Video Streams

Amazon Kinesis Video Streams provides a number of security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don’t represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

For security best practices for your remote devices, see Security Best Practices for Device Agents.

Implement least privilege access

When granting permissions, you decide who is getting what permissions to which Kinesis Video Streams resources. You enable specific actions that you want to allow on those resources. Therefore you should grant only the permissions that are required to perform a task. Implementing least privilege access is fundamental in reducing security risk and the impact that could result from errors or malicious intent.

For example, a producer that sends data to Kinesis Video Streams requires only `PutMedia`, `GetStreamingEndpoint`, and `DescribeStream`. Do not grant producer applications permissions for all actions (*), or for other actions such as `GetMedia`.

For more information, see [What Is Least Privilege & Why Do You Need It?]

Use IAM roles

Producer and client applications must have valid credentials to access Kinesis Video Streams. You should not store AWS credentials directly in a client application or in an Amazon S3 bucket. These are long-term credentials that aren't automatically rotated and could have a significant business impact if they are compromised.

Instead, you should use an IAM role to manage temporary credentials for your producer and client applications to access Kinesis Video Streams. When you use a role, you don't have to use long-term credentials (such as a username and password or access keys) to access other resources.

For more information, see the following topics in the *IAM User Guide*:

- IAM Roles
- Common Scenarios for Roles: Users, Applications, and Services

Use CloudTrail to monitor API calls

Kinesis Video Streams works with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Kinesis Video Streams.

You can use the information collected by CloudTrail to determine the request that was made to Kinesis Video Streams, the IP address from which the request was made, who made the request, when it was made, and additional details.

For more information, see the section called "Logging API Calls with CloudTrail".

Kinesis Video Streams Producer Libraries

The Amazon Kinesis Video Streams Producer libraries are a set of libraries in the Kinesis Video Streams Producer SDK. The client uses the libraries and SDK to build the on-device application for securely connecting to Kinesis Video Streams, and streaming media data to view in the console or client applications in real time.

Media data can be streamed in the following ways:

- In real time
- After buffering it for a few seconds
- After the media uploads

After you create a Kinesis Video Streams stream, you can start sending data to it. You can use the SDK to create application code that extracts the video data, known as frames, from the media source and uploads it to Kinesis Video Streams. These applications are also referred to as *producer* applications.

The Producer libraries contain the following components:

- Kinesis Video Streams Producer Client
- Kinesis Video Streams Producer Library

Kinesis Video Streams Producer Client

The Kinesis Video Streams Producer Client includes a single `KinesisVideoClient` class. This class manages media sources, receives data from the sources, and manages the stream lifecycle as data flows from a media source to Kinesis Video Streams. It also provides a `MediaSource` interface for defining the interaction between Kinesis Video Streams and your proprietary hardware and software.

A media source can be almost anything. For example, you can use a camera media source or a microphone media source. Media sources are not limited to audio and video sources only. For example, data logs might be text files, but they can still be sent as a stream of data. You could also have multiple cameras on your phone that stream data simultaneously.

To get data from any of these sources, you can implement the `MediaSource` interface. This interface enables additional scenarios for which we don’t provide built-in support. For example, you might choose to send the following to Kinesis Video Streams:

- A diagnostic data stream (for example, application logs and events)
- Data from infrared cameras, RADARs, or depth cameras

Kinesis Video Streams doesn't provide built-in implementations for media-producing devices such as cameras. To extract data from these devices, you must implement code, thus creating your own custom media source implementation. You can then explicitly register your custom media sources with `KinesisVideoClient`, which uploads the data to Kinesis Video Streams.

The Kinesis Video Streams Producer Client is available for Java and Android applications. For more information, see Using the Java Producer Library and Using the Android Producer Library.

Kinesis Video Streams Producer Library

The Kinesis Video Streams Producer Library is contained within the Kinesis Video Streams Producer Client. The library is also available to use directly for those who want a deeper integration with Kinesis Video Streams. It enables integration from devices with proprietary operating systems, network stacks, or limited on-device resources.

The Kinesis Video Streams Producer Library implements the state machine for streaming to Kinesis Video Streams. It provides callback hooks, which require that you provide your own transport implementation and explicitly handle each message going to and from the service.

You might choose to use the Kinesis Video Streams Producer Library directly for the following reasons:

- The device on which you want to run the application doesn't have a Java virtual machine.
- You want to write application code in languages other than Java.
- You want to reduce the amount of overhead in your code and limit it to the bare minimum level of abstraction, due to limitations like memory and processing power.

Currently, the Kinesis Video Streams Producer Library is available for Android, C, C++ and Java applications. For more information, see the supported languages in the following *Related Topics*.

Related topics

Using the Java Producer Library

Using the Android Producer Library

Using the C++ Producer Library

Using the C Producer Library

Using the C++ Producer SDK on Raspberry Pi

Using the Java Producer Library

You can use the Amazon Kinesis Video Streams provided Java Producer Library to write application code with minimal configuration, to send media data from a device to a Kinesis video stream.

Perform the following steps to integrate your code with Kinesis Video Streams so that your application can start streaming data to your Kinesis video stream:

1. Create an instance of the `KinesisVideoClient` object.
2. Create a `MediaSource` object by providing media source information. For example, when creating a camera media source, you provide information such as identifying the camera and specifying the encoding the camera uses.

   When you want to start streaming, you must create a custom media source.
3. Register the media source with `KinesisVideoClient`.

   After you register the media source with `KinesisVideoClient`, whenever the data becomes available with the media source, it calls `KinesisVideoClient` with the data.

Procedure: Using the Java Producer SDK

This procedure demonstrates how to use the Kinesis Video Streams Java Producer Client in your Java application to send data to your Kinesis video stream.

These steps don't require you to have a media source, such as a camera or microphone. Instead, for testing purposes, the code generates sample frames that consist of a series of bytes. You can use the same coding pattern when you send media data from real sources such as cameras and microphones.

The procedure includes the following steps:

- Download and Configure the Code
- Write and Examine the Code
- Run and Verify the Code

Prerequisites

- In the sample code, you provide credentials by specifying a profile that you set up in your AWS credentials profile file. If you haven't already done so, first set up your credentials profile. For more information, see Set up AWS Credentials and Region for Development in the *AWS SDK for Java*.

> **Note**
>
> The Java example uses a `SystemPropertiesCredentialsProvider` object to obtain your credentials. The provider retrieves these credentials from the `aws.accessKeyId` and `aws.secretKey` Java system properties. You set these system properties in your Java development environment. For information about how to set Java system properties, see the documentation for your particular integrated development environment (IDE).

- Your `NativeLibraryPath` must contain your `KinesisVideoProducerJNI` file, available at https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp. The file name extension for this file depends on your operating system:
  - **KinesisVideoProducerJNI.so** for Linux
  - **KinesisVideoProducerJNI.dylib** for macOS
  - **KinesisVideoProducerJNI.dll** for Windows

> **Note**
>
> Pre-built libraries for macOS, Ubuntu, Windows, and Raspbian are available in `src/main/resources/lib` at https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-java.git. For other environments, compile the C++ Producer Library.

Step 1: Download and configure the Java Producer Library code

In this section of the Java Producer Library procedure, you download the Java example code, import the project into your Java IDE, and configure the library locations.

For prerequisites and other details about this example, see Using the Java Producer Library.

1. Create a directory, and then clone the example source code from the GitHub repository.

```
git clone https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-java
```

2. Open the Java integrated development environment (IDE) that you use (for example, Eclipse or JetBrains IntelliJ IDEA), and import the Apache Maven project that you downloaded:

   - **In IntelliJ IDEA:** Choose **Import**. Navigate to the `pom.xml` file in the root of the downloaded package.
   - **In Eclipse:** Choose **File**, **Import**, **Maven**, **Existing Maven Projects**. Then navigate to the `kinesis-video-java-demo` directory.

   For more information, see the documentation for your IDE.

3. The Java example code uses the current AWS credentials. To use a different credentials profile, locate the following code in `DemoAppMain.java`:

```
final KinesisVideoClient kinesisVideoClient = KinesisVideoJavaClientFactory
    .createKinesisVideoClient(
        Regions.US_WEST_2,
        AuthHelper.getSystemPropertiesCredentialsProvider());
```

   Change the code to the following:

```
final KinesisVideoClient kinesisVideoClient = KinesisVideoJavaClientFactory
    .createKinesisVideoClient(
        Regions.US_WEST_2,
        new ProfileCredentialsProvider("credentials-profile-name"));
```

   For more information, see ProfileCredentialsProvider in the *AWS SDK for Java* reference.

Next Step

the section called "Step 2: Write and examine the code"

Step 2: Write and examine the code

In this section of the Java Producer Library procedure, you write and examine the Java example code that you downloaded in the previous section.

The Java test application (`DemoAppMain`) shows the following coding pattern:

- Create an instance of `KinesisVideoClient`.
- Create an instance of `MediaSource`.
- Register the `MediaSource` with the client.
- Start streaming. Start the `MediaSource` and it starts sending data to the client.

The following sections provide details.

Creating an instance of KinesisVideoClient

You create the `KinesisVideoClient` object by calling the `createKinesisVideoClient` operation.

```
final KinesisVideoClient kinesisVideoClient = KinesisVideoJavaClientFactory
    .createKinesisVideoClient(
        Regions.US_WEST_2,
        AuthHelper.getSystemPropertiesCredentialsProvider());
```

For `KinesisVideoClient` to make network calls, it needs credentials to authenticate. You pass in an instance of `SystemPropertiesCredentialsProvider`, which reads `AWSCredentials` for the default profile in the credentials file:

```
[default]
aws_access_key_id = ABCDEFGHIJKLMOPQRSTU
aws_secret_access_key = AbCd1234EfGh5678IjKl9012MnOp3456QrSt7890
```

Creating an instance of MediaSource

To send bytes to your Kinesis video stream, you must produce the data. Amazon Kinesis Video Streams provides the `MediaSource` interface, which represents the data source.

For example, the Kinesis Video Streams Java library provides the `ImageFileMediaSource` implementation of the `MediaSource` interface. This class only reads data from a series of media files rather than a Kinesis video stream, but you can use it for testing the code.

```
final MediaSource bytesMediaSource = createImageFileMediaSource();
```

Registering the MediaSource with the client

Register the media source that you created with the `KinesisVideoClient` so that it knows about the client (and can then send data to the client).

```
kinesisVideoClient.registerMediaSource(mediaSource);
```

Starting the media source

Start the media source so that it can begin generating data and send it to the client.

```
bytesMediaSource.start();
```

Next step

the section called “Step 3: Clean up resources”

Step 3: Clean up resources

In order to avoid memory leaks, do the following to unregister a media source from the client and free the client.

```
try {
    kinesisVideoClient.unregisterMediaSource(mediaSource);
    kinesisVideoClient.free();
} catch (final KinesisVideoException e) {
```

```
    throw new RuntimeException(e);
}
```

If you added any items to the cache using the `CachedInfoMultiAuthServiceCallbacks`, for example:

```
serviceCallbacks.addStreamInfoToCache(streamName, streamInfo);
serviceCallbacks.addStreamingEndpointToCache(streamName, dataEndpoint);
```

Clear the cache when you're done:

```
serviceCallbacks.removeStreamFromCache(streamName);
```

Next step

the section called “Step 4: Run and verify the code”

Step 4: Run and verify the code

To run the Java test harness for the Java Producer Library, do the following.

1. Choose **DemoAppMain**.
2. Choose **Run**, **Run 'DemoAppMain'**.
3. Add your credentials to the JVM arguments for the application:
   - **For non-temporary AWS credentials:** `"-Daws.accessKeyId={YourAwsAccessKey} -Daws.secretKey={YourAwsSecretKey} -Djava.library.path={NativeLibraryPath}"`
   - **For temporary AWS credentials:** `"-Daws.accessKeyId={YourAwsAccessKey} -Daws.secretKey={YourAwsSecretKey} -Daws.sessionToken={YourAwsSessionToken} -Djava.library.path={NativeLibraryPath}"`
4. Sign in to the AWS Management Console and open the Kinesis Video Streams console.

   On the **Manage Streams** page, choose your stream.
5. The sample video will play in the embedded player. You might need to wait a short time (up to ten seconds under typical bandwidth and processor conditions) while the frames accumulate before the video appears.

The code example creates a stream. As the `MediaSource` in the code starts, it begins sending sample frames to the `KinesisVideoClient`. The client then sends the data to your Kinesis video stream.

Using the Android Producer Library

You can use the Amazon Kinesis Video Streams provided Android Producer Library to write application code, with minimal configuration, to send media data from an Android device to a Kinesis video stream.

Perform the following steps to integrate your code with Kinesis Video Streams so that your application can start streaming data to your Kinesis video stream:

1. Create an instance of the `KinesisVideoClient` object.
2. Create a `MediaSource` object by providing media source information. For example, when creating a camera media source, you provide information such as identifying the camera and specifying the encoding the camera uses.

   When you want to start streaming, you must create a custom media source.

Procedure: Using the Android Producer SDK

This procedure demonstrates how to use the Kinesis Video Streams Android Producer Client in your Android application to send data to your Kinesis video stream.

The procedure includes the following steps:

- the section called “Prerequisites”
- the section called “Step 1: Download and Configure the Code”
- the section called “Step 2: Examine the code”
- the section called “Step 3: Run and verify the code”

Prerequisites

- We recommend Android Studio for examining, editing, and running the application code. We recommend using the latest stable version.

- In the sample code, you provide Amazon Cognito credentials.

Follow these procedures to set up an Amazon Cognito user pool and identity pool.

- Set up a user pool
- Set up an identity pool

Set up a user pool

To set up a user pool

1. Sign in to the Amazon Cognito console and verify the region is correct.
2. In the navigation on the left choose **User pools**.
3. In the **User pools** section, choose **Create user pool**.
4. Complete the following sections:

    a. **Step 1: Configure sign-in experience** - In the **Cognito user pool sign-in options** section, select the appropriate options.

       Select **Next**.

    b. **Step 2: Configure security requirements** - Select the appropriate options.

       Select **Next**.

    c. **Step 3: Configure sign-up experience** - Select the appropriate options.

       Select **Next**.

    d. **Step 4: Configure message delivery** - Select the appropriate options.

       In the **IAM role selection** field, select an existing role or create a new role.

       Select **Next**.

    e. **Step 5: Integrate your app** - Select the appropriate options.

       In the **Initial app client** field, choose **Confidential client**.

       Select **Next**.

    f. **Step 6: Review and create** - Review your selections from the previous sections, then choose **Create user pool**.

5. On the **User pools** page, select the pool that you just created.

   Copy the **User pool ID** and make note of this for later. In the `awsconfiguration.json` file, this is `CognitoUserPool.Default.PoolId`.
6. Select the **App integration** tab and go to the bottom of the page.
7. In the **App client list** section, choose the **App client name** you just created.

   Copy the **Client ID** and make note of this for later. In the `awsconfiguration.json` file, this is `CognitoUserPool.Default.AppClientId`.
8. Show the **Client secret** and make note of this for later. In the `awsconfiguration.json` file, this is `CognitoUserPool.Default.AppClientSecret`.

Set up an identity pool

To set up an identity pool

1. Sign in to the [Amazon Cognito console](#) and verify the region is correct.
2. In the navigation on the left choose **Identity pools**.
3. Choose **Create identity pool**.
4. Configure the identity pool.

   a. **Step 1: Configure identity pool trust** - Complete the following sections:

      - **User access** - Select **Authenticated access**
      - **Authenticated identity sources** - Select **Amazon Cognito user pool**

      Select **Next**.

   b. **Step 2: Configure permissions** - In the **Authenticated role** section, complete the following fields:

      - **IAM role** - Select **Create a new IAM role**
      - **IAM role name** - Enter a name and make note of it for a later step.

      Select **Next**.

   c. **Step 3: Connect identity providers** - In the **User pool details** section complete the following fields:

- **User pool ID** - Select the user pool you created earlier.
- **App client ID** - Select the app client ID you created earlier.

Select **Next**.

d. **Step 4: Configure properties** - Type a name in the **Identity pool name** field.

Select **Next**.

e. **Step 5: Review and create** - Review your selections in each of the sections, then select **Create identity pool**.

5. On the **Identity pools** page, select your new identity pool.

Copy the **Identity pool ID** and make note of this for later. In the `awsconfiguration.json` file, this is `CredentialsProvider.CognitoIdentity.Default.PoolId`.

6. Update the permissions for the IAM role.

a. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

b. In the navigation on the left, choose **Roles**.

c. Find and select the role you created above.

> **Note**
>
> Use the search bar, if needed.

d. Select the attached permissions policy.

Select **Edit**.

e. Select the **JSON** tab and replace the policy with the following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cognito-identity:*",
                "kinesisvideo:*"
```

```
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Select **Next**.

f. Select the box next to **Set this new version as the default** if it isn't already selected.

   Select **Save changes**.

Step 1: Download and configure the Android Producer Library code

In this section of the Android Producer Library procedure, you download the Android example code and open the project in Android Studio.

For prerequisites and other details about this example, see Using the Android Producer Library.

1. Create a directory, and then clone the AWS Mobile SDK for Android from the GitHub repository.

```
git clone https://github.com/awslabs/aws-sdk-android-samples
```

2. Open Android Studio.
3. In the opening screen, choose **Open an existing Android Studio project**.
4. Navigate to the `aws-sdk-android-samples/AmazonKinesisVideoDemoApp` directory, and choose **OK**.
5. Open the `AmazonKinesisVideoDemoApp/src/main/res/raw/awsconfiguration.json` file.

   In the `CredentialsProvider` node, provide the identity pool ID from the **To set up an identity pool** procedure in the Prerequisites section, and provide your AWS Region (for example, **`us-west-2`**).

In the `CognitoUserPool` node, provide the App client secret, App client ID, and Pool ID from the **To set up a user pool** procedure in the Prerequisites section, and provide your AWS Region (for example, **`us-west-2`**).

6. Your `awsconfiguration.json` file will look similar to the following:

```
{
  "Version": "1.0",
  "CredentialsProvider": {
    "CognitoIdentity": {
      "Default": {
        "PoolId": "us-west-2:01234567-89ab-cdef-0123-456789abcdef",
        "Region": "us-west-2"
      }
    }
  },
  "IdentityManager": {
    "Default": {}
  },
  "CognitoUserPool": {
    "Default": {
      "AppClientSecret": "abcdefghijklmnopqrstuvwxyz0123456789abcdefghijklmno",
      "AppClientId": "0123456789abcdefghijklmnop",
      "PoolId": "us-west-2_qRsTuVwXy",
      "Region": "us-west-2"
    }
  }
}
```

7. Update the `AmazonKinesisVideoDemoApp/src/main/java/com/amazonaws/kinesisvideo/demoapp/KinesisVideoDemoApp.java` with your Region (in the following sample, it's set to **US_WEST_2**):

```
public class KinesisVideoDemoApp extends Application {
    public static final String TAG = KinesisVideoDemoApp.class.getSimpleName();
    public static Regions KINESIS_VIDEO_REGION = Regions.US_WEST_2;
```

For information about AWS Region constants, see Regions.

Next step

the section called "Step 2: Examine the code"

Step 2: Examine the code

In this section of the Android Producer Library procedure, you examine the example code.

The Android test application (`AmazonKinesisVideoDemoApp`) shows the following coding pattern:

- Create an instance of `KinesisVideoClient`.
- Create an instance of `MediaSource`.
- Start streaming. Start the `MediaSource`, and it starts sending data to the client.

The following sections provide details.

Creating an instance of KinesisVideoClient

You create the `KinesisVideoClient` object by calling the `createKinesisVideoClient` operation.

```
mKinesisVideoClient = KinesisVideoAndroidClientFactory.createKinesisVideoClient(
                    getActivity(),
                    KinesisVideoDemoApp.KINESIS_VIDEO_REGION,
                    KinesisVideoDemoApp.getCredentialsProvider());
```

For `KinesisVideoClient` to make network calls, it needs credentials to authenticate. You pass in an instance of `AWSCredentialsProvider`, which reads your Amazon Cognito credentials from the `awsconfiguration.json` file that you modified in the previous section.

Creating an instance of MediaSource

To send bytes to your Kinesis video stream, you must produce the data. Amazon Kinesis Video Streams provides the `MediaSource` interface, which represents the data source.

For example, the Kinesis Video Streams Android library provides the `AndroidCameraMediaSource` implementation of the `MediaSource` interface. This class reads data from one of the device's cameras.

In the following code example (from the [fragment/StreamConfigurationFragment.java]() file), the configuration for the media source is created:

```
private AndroidCameraMediaSourceConfiguration getCurrentConfiguration() {
return new AndroidCameraMediaSourceConfiguration(
        AndroidCameraMediaSourceConfiguration.builder()
                .withCameraId(mCamerasDropdown.getSelectedItem().getCameraId())

 .withEncodingMimeType(mMimeTypeDropdown.getSelectedItem().getMimeType())

 .withHorizontalResolution(mResolutionDropdown.getSelectedItem().getWidth())

 .withVerticalResolution(mResolutionDropdown.getSelectedItem().getHeight())
                .withCameraFacing(mCamerasDropdown.getSelectedItem().getCameraFacing())
                .withIsEncoderHardwareAccelerated(

 mCamerasDropdown.getSelectedItem().isEndcoderHardwareAccelerated())
                .withFrameRate(FRAMERATE_20)
                .withRetentionPeriodInHours(RETENTION_PERIOD_48_HOURS)
                .withEncodingBitRate(BITRATE_384_KBPS)
                .withCameraOrientation(-
mCamerasDropdown.getSelectedItem().getCameraOrientation())

 .withNalAdaptationFlags(StreamInfo.NalAdaptationFlags.NAL_ADAPTATION_ANNEXB_CPD_AND_FRAME_NALS
                .withIsAbsoluteTimecode(false));
}
```

In the following code example (from the [fragment/StreamingFragment.java]() file), the media source is created:

```
mCameraMediaSource = (AndroidCameraMediaSource) mKinesisVideoClient
    .createMediaSource(mStreamName, mConfiguration);
```

Starting the media source

Start the media source so that it can begin generating data and sending it to the client. The following code example is from the [fragment/StreamingFragment.java]() file:

```
mCameraMediaSource.start();
```

Next step

the section called "Step 3: Run and verify the code"

Step 3: Run and verify the code

To run the Android example application for the Android Producer Library, do the following.

1. Connect to an Android device.
2. Choose **Run**, **Run...**, and choose **Edit configurations...**.
3. Choose the plus icon (**+**), **Android App**. In the **Name** field, enter **AmazonKinesisVideoDemoApp**. In the **Module** pulldown, choose **AmazonKinesisVideoDemoApp**. Choose **OK**.
4. Choose **Run**, **Run**.
5. In the **Select Deployment Target** screen, choose your connected device, and choose **OK**.
6. In the **AWSKinesisVideoDemoApp** application on the device, choose **Create new account**.
7. Enter values for **USERNAME**, **Password**, **Given name**, **Email address**, and **Phone number**, and then choose **Sign up**.

   > **Note**
   >
   > These values have the following constraints:
   >
   > - **Password:** Must contain uppercase and lowercase letters, numbers, and special characters. You can change these constraints in your User pool page on the Amazon Cognito console.
   > - **Email address:** Must be a valid address so that you can receive a confirmation code.
   > - **Phone number:** Must be in the following format: **+*<Country code><Number>***, for example, **+12065551212**.

8. Enter the code that you receive by email, and choose **Confirm**. Choose **Ok**.
9. On the next page, keep the default values, and choose **Stream**.
10. Sign in to the AWS Management Console and open the Kinesis Video Streams console in the US West (Oregon) Region.

    On the **Manage Streams** page, choose **demo-stream**.

11. The streaming video plays in the embedded player. You might need to wait a short time (up to ten seconds under typical bandwidth and processor conditions) while the frames accumulate before the video appears.

> **Note**
>
> If the device's screen rotates (for example, from portrait to landscape), the application stops streaming video.

The code example creates a stream. As the `MediaSource` in the code starts, it begins sending frames from the camera to the `KinesisVideoClient`. The client then sends the data to a Kinesis video stream named **demo-stream**.

Using the C++ Producer Library

You can use the Amazon Kinesis Video Streams provided C++ Producer Library to write application code to send media data from a device to a Kinesis video stream.

Object model

The C++ library provides the following objects to manage sending data to a Kinesis video stream:

- **KinesisVideoProducer:** Contains information about your media source and AWS credentials, and maintains callbacks to report on Kinesis Video Streams events.
- **KinesisVideoStream:** Represents the Kinesis video stream. Contains information about the video stream's parameters, such as name, data retention period, and media content type.

Putting media into the stream

You can use the C++ library provided methods (for example, `PutFrame`) to put data into the `KinesisVideoStream` object. The library then manages the internal state of the data, which can include the following tasks:

- Performing authentication.
- Watching for network latency. If the latency is too high, the library might choose to drop frames.
- Tracking status of streaming in progress.

Callback interfaces

This layer exposes a set of callback interfaces, which enable it to talk to the application layer. These callback interfaces include the following:

- Service callbacks interface (`CallbackProvider`): The library invokes events obtained through this interface when it creates a stream, obtains a stream description, and deletes a stream.
- Client-ready state or low storage events interface (`ClientCallbackProvider`): The library invokes events on this interface when the client is ready, or when it detects that it might run out of available storage or memory.
- Stream events callback interface (`StreamCallbackProvider`): The library invokes events on this interface when stream events occur, such as the stream entering the ready state, dropped frames, or stream errors.

Kinesis Video Streams provides default implementations for these interfaces. You can also provide your own custom implementation—for example, if you need custom networking logic or you want to expose a low storage condition to the user interface.

For more information about callbacks in the Producer Libraries, see Producer SDK callbacks.

Procedure: Using the C++ Producer SDK

This procedure demonstrates how to use the Kinesis Video Streams client and media sources in a C ++ application to send data to your Kinesis video stream.

The procedure includes the following steps:

- Step 1: Download and Configure the Code
- Step 2: Write and Examine the Code
- Step 3: Run and Verify the Code

Prerequisites

- **Credentials:** In the sample code, you provide credentials by specifying a profile that you set up in your AWS credentials profile file. If you haven't already done so, first set up your credentials profile.

For more information, see Set up AWS Credentials and Region for Development.

- **Certificate store integration:** The Kinesis Video Streams Producer Library must establish trust with the service it calls. This is done through validating the certificate authorities (CAs) in the public certificate store. On Linux-based models, this store is located in the `/etc/ssl/` directory.

  Download the certificate from the following location to your certificate store:

  https://www.amazontrust.com/repository/SFSRootCAG2.pem
- Install the following build dependencies for macOS:
  - Autoconf 2.69 (License GPLv3+/Autoconf: GNU GPL version 3 or later)
  - CMake 3.7 or 3.8
  - Pkg-Config
  - xCode (macOS) / clang / gcc (xcode-select version 2347)
  - Java Development Kit (JDK) (for Java JNI compilation)
  - Lib-Pkg
- Install the following build dependencies for Ubuntu:
  - Git: `sudo apt install git`
  - CMake: `sudo apt install cmake`
  - G++: `sudo apt install g++`
  - pkg-config: `sudo apt install pkg-config`
  - OpenJDK: `sudo apt install openjdk-8-jdk`

    Note

    This is only required if you're building Java Native Interface (JNI).
  - Set the `JAVA_HOME` environment variable: `export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/`

Next step

Step 1: Download and Configure the C++ Producer Library Code

Step 1: Download and configure the C++ Producer Library code

For information about how to download and configure the C++ Producer Library, see Amazon Kinesis Video Streams CPP Producer, GStreamer Plugin and JNI.

For prerequisites and more information about this example, see Using the C++ Producer Library.

Next step

Step 2: Write and examine the code

Step 2: Write and examine the code

In this section of the C++ Producer Library procedure, you examine the code in the C++ test harness (`tst/ProducerTestFixture.h` and other files). You downloaded this code in the previous section.

The **Platform Independent** C++ example shows the following coding pattern:

- Create an instance of `KinesisVideoProducer` to access Kinesis Video Streams.
- Create an instance of `KinesisVideoStream`. This creates a Kinesis video stream in your AWS account if a stream of the same name doesn't already exist.
- Call `putFrame` on the `KinesisVideoStream` for every frame of data, as it becomes available, to send it to the stream.

The following sections provide more information about this coding pattern.

Creating an instance of KinesisVideoProducer

You create the `KinesisVideoProducer` object by calling the `KinesisVideoProducer::createSync` method. The following example creates the `KinesisVideoProducer` in the `ProducerTestFixture.h` file:

```
kinesis_video_producer_ = KinesisVideoProducer::createSync(move(device_provider_),
    move(client_callback_provider_),
    move(stream_callback_provider_),
    move(credential_provider_),
    defaultRegion_);
```

The `createSync` method takes the following parameters:

- A `DeviceInfoProvider` object, which returns a `DeviceInfo` object containing information about the device or storage configuration.

> **Note**
>
> You configure your content store size using the `deviceInfo.storageInfo.storageSize` parameter. Your content streams share the content store. To determine your storage size requirement, multiply the average frame size by the number of frames stored for the max duration for all the streams. Then multiply by 1.2 to account for defragmentation. For example, suppose that your application has the following configuration:
>
> - Three streams
> - 3 minutes of maximum duration
> - Each stream is 30 frames per second (FPS)
> - Each frame is 10,000 KB in size
>
> The content store requirement for this application is **3 (streams) * 3 (minutes) * 60 (seconds in a minute) * 10000 (kb) * 1.2 (defragmentation allowance) = 194.4 Mb ~ 200 Mb**.

- A `ClientCallbackProvider` object, which returns function pointers that report client-specific events.
- A `StreamCallbackProvider` object, which returns function pointers that are called back when stream-specific events occur.
- A `CredentialProvider` object, which provides access to AWS credential environment variables.
- The AWS Region ("us-west-2"). The service endpoint is determined from the Region.

Creating an instance of KinesisVideoStream

You create the `KinesisVideoStream` object by calling the `KinesisVideoProducer::CreateStream` method with a `StreamDefinition` parameter. The example creates the `KinesisVideoStream` in the `ProducerTestFixture.h` file with the track type as video, and with track id as 1:

```
auto stream_definition = make_unique<StreamDefinition>(stream_name,
                                                       hours(2),
```

```
                                                        tags,
                                                        "",
                                                        STREAMING_TYPE_REALTIME,
                                                        "video/h264",
                                                        milliseconds::zero(),
                                                        seconds(2),
                                                        milliseconds(1),
                                                        true,
                                                        true,
                                                        true);
return kinesis_video_producer_->createStream(move(stream_definition));
```

The `StreamDefinition` object has the following fields:

- Stream name.
- Data retention period.
- Tags for the stream. These tags can be used by consumer applications to find the correct stream, or to get more information about the stream. The tags can also be viewed in the AWS Management Console.
- AWS KMS encryption key for the stream. For more information, see Using Server-Side Encryption with Kinesis Video Streams.
- Streaming type. Currently, the only valid value is `STREAMING_TYPE_REALTIME`.
- Media content type.
- Media latency. This value isn't currently used, and should be set to 0.
- Playback duration of each fragment.
- Media timecode scale.
- Whether the media uses key frame fragmentation.
- Whether the media uses timecodes.
- Whether the media uses absolute fragment times.

Adding an audio track to the Kinesis video stream

You can add audio track details to a video track stream definition by using the addTrack method of the StreamDefinition:

```
stream_definition->addTrack(DEFAULT_AUDIO_TRACKID, DEFAULT_AUDIO_TRACK_NAME,
 DEFAULT_AUDIO_CODEC_ID, MKV_TRACK_INFO_TYPE_AUDIO);
```

The addTrack method requires the following parameters:

- Track id (as one for audio). This should be unique and non-zero value.
- User-defined track name (for example, "audio" for the audio track).
- Codec id for this track (for example, for audio track "A_AAC").
- Track type (for example, use the enum value of MKV_TRACK_INFO_TYPE_AUDIO for audio).

If you have codec private data for the audio track, then you can pass it when calling the addTrack function. You can also send the codec private data after creating the KinesisVideoStream object while calling the start method in KinesisVideoStream.

Putting a frame into the Kinesis video stream

You put media into the Kinesis video stream using `KinesisVideoStream::putFrame`, passing in a `Frame` object that contains the header and media data. The example calls `putFrame` in the `ProducerApiTest.cpp` file:

```
frame.duration = FRAME_DURATION_IN_MICROS * HUNDREDS_OF_NANOS_IN_A_MICROSECOND;
    frame.size = SIZEOF(frameBuffer_);
    frame.frameData = frameBuffer_;
    MEMSET(frame.frameData, 0x55, frame.size);

    while (!stop_producer_) {
        // Produce frames
        timestamp = std::chrono::duration_cast<std::chrono::nanoseconds>(
                std::chrono::system_clock::now().time_since_epoch()).count() /
 DEFAULT_TIME_UNIT_IN_NANOS;
        frame.index = index++;
        frame.decodingTs = timestamp;
        frame.presentationTs = timestamp;

        // Key frame every 50th
        frame.flags = (frame.index % 50 == 0) ? FRAME_FLAG_KEY_FRAME : FRAME_FLAG_NONE;
    ...

    EXPECT_TRUE(kinesis_video_stream->putFrame(frame));
```

> **Note**
>
> The preceding C++ Producer example sends a buffer of test data. In a real-world application, you should obtain the frame buffer and size from the frame data from a media source (such as a camera).

The `Frame` object has the following fields:

- Frame index. This should be a monotonically incrementing value.
- Flags associated with the frame. For example, if the encoder were configured to produce a key frame, this frame would be assigned the `FRAME_FLAG_KEY_FRAME` flag.
- Decoding timestamp.
- Presentation timestamp.
- Duration of the frame (to 100 ns units).
- Size of the frame in bytes.
- Frame data.

For more information about the format of the frame, see Kinesis Video Streams Data Model.

Putting a KinesisVideoFrame into a specific track of KinesisVideoStream

You can use the PutFrameHelper class to put frame data into a specific track. First, call the getFrameDataBuffer to get a pointer to one of the pre-allocated buffers to fill in the KinesisVideoFrame data. Then, you can call the putFrameMultiTrack to send the KinesisVideoFrame along with the Boolean value to indicate the type of frame data. Use true if it's a video data or false if the frame contains audio data. The putFrameMultiTrack method uses a queueing mechanism to ensure that the MKV Fragments maintain monotonically increasing frame timestamps and any two fragments don't overlap. For example, the MKV timestamp of the first frame of a fragment should always be greater than the MKV timestamp of the last frame of the previous fragment.

The PutFrameHelper has the following fields:

- Maximum number of audio frames in the queue.
- Maximum number of video frames in the queue.
- Size to allocate for a single audio frame.

- Size to allocate for a single video frame.

Metrics and metric logging

The C++ Producer SDK includes functionality for metrics and metric logging.

You can use the `getKinesisVideoMetrics` and `getKinesisVideoStreamMetrics` API operations to retrieve information about Kinesis Video Streams and your active streams.

The following code is from the `kinesis-video-pic/src/client/include/com/amazonaws/kinesis/video/client/Include.h` file.

```
/**
* Gets information about the storage availability.
*
* @param 1 CLIENT_HANDLE - the client object handle.
* @param 2 PKinesisVideoMetrics - OUT - Kinesis Video metrics to be filled.
*
* @return Status of the function call.
*/
PUBLIC_API STATUS getKinesisVideoMetrics(CLIENT_HANDLE, PKinesisVideoMetrics);

/**
* Gets information about the stream content view.
*
* @param 1 STREAM_HANDLE - the stream object handle.
* @param 2 PStreamMetrics - Stream metrics to fill.
*
* @return Status of the function call.
*/
PUBLIC_API STATUS getKinesisVideoStreamMetrics(STREAM_HANDLE, PStreamMetrics);
```

The `PClientMetrics` object filled by `getKinesisVideoMetrics` contains the following information:

- **contentStoreSize:** The overall size in bytes of the content store (the memory used to store streaming data).
- **contentStoreAvailableSize:** The available memory in the content store, in bytes.
- **contentStoreAllocatedSize:** The allocated memory in the content store.

- **totalContentViewsSize:** The total memory used for the content view. The content view is a series of indices of information in the content store.
- **totalFrameRate:** The aggregate number of frames per second across all active streams.
- **totalTransferRate:** The total bits per second (bps) being sent in all streams.

The `PStreamMetrics` object filled by `getKinesisVideoStreamMetrics` contains the following information:

- **currentViewDuration:** The difference in 100 ns units between the head of the content view (when frames are encoded) and the current position (when frame data is sent to Kinesis Video Streams).
- **overallViewDuration:** The difference in 100 ns units between the head of the content view (when frames are encoded) to the tail (when frames are flushed from memory, either because the total allocated space for the content view is exceeded, or because a `PersistedAck` message is received from Kinesis Video Streams, and frames known to be persisted are flushed).
- **currentViewSize:** The size in bytes of the content view from the head (when frames are encoded) to the current position (when frames are sent to Kinesis Video Streams).
- **overallViewSize:** The total size in bytes of the content view.
- **currentFrameRate:** The last measured rate of the stream, in frames per second.
- **currentTransferRate:** The last measured rate of the stream, in bytes per second.

Teardown

If you want to send the remaining bytes in a buffer and wait for the ACK, you can use `stopSync`:

```
kinesis_video_stream->stopSync();
```

Or you can call `stop` to end the streaming:

```
kinesis_video_stream->stop();
```

After stopping the stream, you can free the stream through invoking the following API:

```
kinesis_video_producer_->freeStream(kinesis_video_stream);
```

Next step

the section called “Step 3: Run and verify the code”

Step 3: Run and verify the code

To run and verify the code for the C++ Producer Library procedure, see the following OS-specific instructions:

- Linux
- macOS
- Windows
- Raspberry Pi OS

You can monitor the traffic on your stream by watching the metrics that are associated with your stream in the Amazon CloudWatch console, such as `PutMedia.IncomingBytes`.

Using the C++ Producer SDK as a GStreamer plugin

GStreamer is a popular media framework used by multiple cameras and video sources to create custom media pipelines by combining modular plugins. The Kinesis Video Streams GStreamer plugin streamlines the integration of your existing GStreamer media pipeline with Kinesis Video Streams.

For information about using the C++ Producer SDK as a GStreamer plugin, see Example: Kinesis Video Streams Producer SDK GStreamer Plugin - kvssink.

Using the C++ Producer SDK as a GStreamer plugin in a Docker container

GStreamer is a popular media framework used by multiple cameras and video sources to create custom media pipelines by combining modular plugins. The Kinesis Video Streams GStreamer plugin streamlines the integration of your existing GStreamer media pipeline with Kinesis Video Streams.

In addition, using [Docker] to create the GStreamer pipeline standardizes the operating environment for Kinesis Video Streams, which streamlines building and running the application.

For information about using the C++ Producer SDK as a GStreamer plugin in a Docker container, see [Run the GStreamer element in a Docker container].

Using logging with the C++ Producer SDK

You configure logging for C++ Producer SDK applications in the `kvs_log_configuration` file in the `kinesis-video-native-build` folder.

The following example shows the first line of the default configuration file, which configures the application to write DEBUG-level log entries to the AWS Management Console:

```
log4cplus.rootLogger=DEBUG, KvsConsoleAppender
```

You can set the logging level to `INFO` for less verbose logging.

To configure the application to write log entries to a log file, update the first line in the file to the following:

```
log4cplus.rootLogger=DEBUG, KvsConsoleAppender, KvsFileAppender
```

This configures the application to write log entries to `kvs.log` in the `kinesis-video-native-build/log` folder.

To change the log file location, update the following line with the new path:

```
log4cplus.appender.KvsFileAppender.File=./log/kvs.log
```

> **Note**
>
> If DEBUG-level logging is written to a file, the log file can use up the available storage space on the device quickly.

Using the C Producer Library

You can use the Amazon Kinesis Video Streams provided C Producer Library to write application code to send media data from a device to a Kinesis video stream.

Object model

The Kinesis Video Streams C Producer Library is based on a common component called Platform Independent Codebase (PIC), which is available on GitHub at https://github.com/awslabs/amazon-kinesis-video-streams-pic/. The PIC contains platform-independent business logic for the foundational components. The Kinesis Video Streams C Producer Library wraps PIC with additional layer of API that allows scenario-and platform-specific callbacks and events. The Kinesis Video Streams C Producer Library has the following components built on top of PIC:

- **Device info providers** – Exposes the `DeviceInfo` structure that can be directly supplied to the PIC API. You can configure a set of providers, including application scenario-optimized provider that can optimize the content store based on the number and types of streams that your application handles and the amount of required buffering configured based on the amount of available RAM.
- **Stream info provider** – Exposes the `StreamInfo` structure that can be directly supplied to the PIC API. There's a set of providers that are specific to the application types and the common types of streaming scenarios. These include providers such as video, audio, and audio and video multitrack. Each of these scenarios have defaults that you can customize according to your application's requirements.
- **Callback provider** – Exposes the `ClientCallbacks` structure that can be directly supplied to the PIC API. This includes a set of callback providers for networking (CURL-based API callbacks), authorization (AWS credentials API), and retry streaming on errors callbacks. The Callback Provider API takes a number of arguments to configure, such as the AWS Region and authorization information. This is done by using IoT certificates or by using AWS AccessKeyId, SecretKey, or SessionToken. You can enhance Callback Provider with custom callbacks if your application needs further processing of a particular callback to achieve some application-specific logic.
- **FrameOrderCoordinator** – Helps handle audio and video synchronization for multi-track scenarios. It has default behavior, which you can customize to handle your application's specific logic. It also streamlines the frame metadata packaging in PIC Frame structure before submitting it to the lower-layer PIC API. For non-multitrack scenarios, this component is a pass-through to PIC putFrame API.

The C library provides the following objects to manage sending data to a Kinesis video stream:

- **KinesisVideoClient** – Contains information about your device and maintains callbacks to report on Kinesis Video Streams events.
- **KinesisVideoStream** – Represents information about the video stream's parameters, such as name, data retention period, and media content type.

Putting media into the stream

You can use the C library provided methods (for example, `PutKinesisVideoFrame`) to put data into the `KinesisVideoStream` object. The library then manages the internal state of the data, which can include the following tasks:

- Performing authentication.
- Watching for network latency. If the latency is too high, the library might choose to drop frames.
- Tracking status of streaming in progress.

Procedure: Using the C Producer SDK

This procedure demonstrates how to use the Kinesis Video Streams client and media sources in a C application to send H.264-encoded video frames to your Kinesis video stream.

The procedure includes the following steps:

- Step 1: Download the C Producer Library code
- Step 2: Write and examine the code
- Step 3: Run and verify the code

Prerequisites

- **Credentials** – In the sample code, you provide credentials by specifying a profile that you set up in your AWS credentials profile file. If you haven't already done so, first set up your credentials profile.

  For more information, see Set up AWS Credentials and Region for Development.

- **Certificate store integration** – The Kinesis Video Streams Producer Library must establish trust with the service it calls. This is done through validating the certificate authorities (CAs) in the public certificate store. On Linux-based models, this store is located in the `/etc/ssl/` directory.

  Download the certificate from the following location to your certificate store:

  https://www.amazontrust.com/repository/SFSRootCAG2.pem
- Install the following build dependencies for macOS:
  - Autoconf 2.69 (License GPLv3+/Autoconf: GNU GPL version 3 or later)
  - CMake 3.7 or 3.8
  - Pkg-Config
  - xCode (macOS) / clang / gcc (xcode-select version 2347)
  - Java Development Kit (JDK) (for Java JNI compilation)
  - Lib-Pkg
- Install the following build dependencies for Ubuntu:
  - Git: `sudo apt install git`
  - CMake: `sudo apt install cmake`
  - G++: `sudo apt install g++`
  - pkg-config: `sudo apt install pkg-config`
  - OpenJDK: `sudo apt install openjdk-8-jdk`
  - Set the `JAVA_HOME` environment variable: `export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/`

Next step

Step 1: Download the C Producer Library code

Step 1: Download the C Producer Library code

In this section, you download the low-level libraries. For prerequisites and other details about this example, see Using the C Producer Library.

1. Create a directory, and then clone the example source code from the GitHub repository.

```
git clone --recursive https://github.com/awslabs/amazon-kinesis-video-streams-
producer-c.git
```

> **Note**
>
> If you miss running git clone with `--recursive`, run `git submodule update --init` in the `amazon-kinesis-video-streams-producer-c/open-source` directory. You must also install pkg-config, CMake, and a build environment.
> For more information, see the `README.md` in https://github.com/awslabs/amazon-kinesis-video-streams-producer-c.git.

2. Open the code in the integrated development environment (IDE) of your choice (for example, Eclipse).

Next step

Step 2: Write and examine the code

Step 2: Write and examine the code

In this section, you examine the code of the sample application `KvsVideoOnlyStreamingSample.c` in the `samples` folder of the https://github.com/awslabs/amazon-kinesis-video-streams-producer-c repo on GitHub. You downloaded this code in the previous step. This sample demonstrates how to use the C Producer Library to send H.264-encoded video frames inside the folder `samples/h264SampleFrames` to your Kinesis video stream.

This sample application has three sections:

- Initialization and configuration:
  - Initializing and configuring the platform-specific media pipeline.
  - Initializing and configuring KinesisVideoClient and KinesisVideoStream for the pipeline, setting the callbacks, integrating scenario-specific authentication, extracting and submitting codec private data, and getting the stream to READY state.
- Main loop:
  - Getting the frame from the media pipeline with the timestamps and flags.
  - Submitting the frame to the KinesisVideoStream.

- Teardown:
  - Stopping (sync) KinesisVideoStream, freeing KinesisVideoStream, freeing KinesisVideoClient.

This sample application completes the following tasks:

- Call the `createDefaultDeviceInfo` API to create the `deviceInfo` object that contains information about the device or storage configuration.

```
// default storage size is 128MB. Use setDeviceInfoStorageSize after create to change
 storage size.
CHK_STATUS(createDefaultDeviceInfo(&pDeviceInfo));
// adjust members of pDeviceInfo here if needed
    pDeviceInfo->clientInfo.loggerLogLevel = LOG_LEVEL_DEBUG;
```

- Call the `createRealtimeVideoStreamInfoProvider` API to create the `StreamInfo` object.

```
CHK_STATUS(createRealtimeVideoStreamInfoProvider(streamName,
 DEFAULT_RETENTION_PERIOD, DEFAULT_BUFFER_DURATION, &pStreamInfo));
// adjust members of pStreamInfo here if needed
```

- Call the `createDefaultCallbacksProviderWithAwsCredentials` API to create the default callbacks provider based on static AWS credentials.

```
CHK_STATUS(createDefaultCallbacksProviderWithAwsCredentials(accessKey,
                                                                secretKey,
                                                                sessionToken,
                                                                MAX_UINT64,
                                                                region,
                                                                cacertPath,
                                                                NULL,
                                                                NULL,
                                                                FALSE,
                                                                &pClientCallbacks));
```

- Call the `createKinesisVideoClient` API to create the `KinesisVideoClient` object that contains information about your device storage and maintains callbacks to report on Kinesis Video Streams events.

```
CHK_STATUS(createKinesisVideoClient(pDeviceInfo, pClientCallbacks, &clientHandle));
```

- Call the `createKinesisVideoStreamSync` API to create the `KinesisVideoStream` object.

```
CHK_STATUS(createKinesisVideoStreamSync(clientHandle, pStreamInfo, &streamHandle));
```

- Set up a sample frame and call `PutKinesisVideoFrame` API to send that frame to the `KinesisVideoStream` object.

```
 // setup sample frame
    MEMSET(frameBuffer, 0x00, frameSize);
    frame.frameData = frameBuffer;
    frame.version = FRAME_CURRENT_VERSION;
    frame.trackId = DEFAULT_VIDEO_TRACK_ID;
    frame.duration = HUNDREDS_OF_NANOS_IN_A_SECOND / DEFAULT_FPS_VALUE;
    frame.decodingTs = defaultGetTime(); // current time
    frame.presentationTs = frame.decodingTs;

    while(defaultGetTime() > streamStopTime) {
        frame.index = frameIndex;
        frame.flags = fileIndex % DEFAULT_KEY_FRAME_INTERVAL == 0 ?
 FRAME_FLAG_KEY_FRAME : FRAME_FLAG_NONE;
        frame.size = SIZEOF(frameBuffer);

        CHK_STATUS(readFrameData(&frame, frameFilePath));

        CHK_STATUS(putKinesisVideoFrame(streamHandle, &frame));
        defaultThreadSleep(frame.duration);

        frame.decodingTs += frame.duration;
        frame.presentationTs = frame.decodingTs;
        frameIndex++;
        fileIndex++;
```

```
        fileIndex = fileIndex % NUMBER_OF_FRAME_FILES;
    }
```

- Teardown:

```
CHK_STATUS(stopKinesisVideoStreamSync(streamHandle));
CHK_STATUS(freeKinesisVideoStream(&streamHandle));
CHK_STATUS(freeKinesisVideoClient(&clientHandle));
```

Next step

Step 3: Run and verify the code

Step 3: Run and verify the code

To run and verify the code for the Producer Library procedure, do the following:

1. Run the following commands to create a build directory in your downloaded C SDK, and launch cmake from it:

```
mkdir -p amazon-kinesis-video-streams-producer-c/build;
cd amazon-kinesis-video-streams-producer-c/build;
cmake ..
```

   You can pass the following options to cmake ..

   - -DBUILD_DEPENDENCIES - whether to build depending libraries from source.
   - -DBUILD_TEST=TRUE - build unit and integration tests. Might be useful to confirm support for your device.

     ./tst/webrtc_client_test
   - -DCODE_COVERAGE - enable coverage reporting.
   - -DCOMPILER_WARNINGS - enable all compiler warnings.

  - `-DADDRESS_SANITIZER` - build with AddressSanitizer.
  - `-DMEMORY_SANITIZER` - build with MemorySanitizer.
  - `-DTHREAD_SANITIZER` - build with ThreadSanitizer.
  - `-DUNDEFINED_BEHAVIOR_SANITIZER` - build with UndefinedBehaviorSanitizer.
  - `-DALIGNED_MEMORY_MODEL` - build for aligned memory model only devices. Default is `OFF`.

2. Navigate to the `build` directory that you just created with the previous step, and run `make` to build the WebRTC C SDK and its provided samples.

```
make
```

3. The sample application `kinesis_video_cproducer_video_only_sample` sends h.264-encoded video frames inside the folder `samples/h264SampleFrames` to Kinesis Video Streams. The following command sends the video frames in a loop for ten seconds to Kinesis Video Streams:

```
./kinesis_video_cproducer_video_only_sample YourStreamName 10
```

   If you want to send H.264-encoded frames from another folder (for example, `MyH264FramesFolder`), run the sample with the following arguments:

```
./kinesis_video_cproducer_video_only_sample YourStreamName 10 MyH264FramesFolder
```

4. To enable verbose logs, define the `HEAP_DEBUG` and `LOG_STREAMING` C-defines by uncommenting the appropriate lines in `CMakeList.txt`.

You can monitor the progress of the test suite in the debug output in your IDE. You can also monitor the traffic on your stream by watching the metrics that are associated with your stream in the Amazon CloudWatch console, such as `PutMedia.IncomingBytes`.

> **Note**
>
> The console doesn't display the data as a video stream because the test harness only sends frames of empty bytes.

Using the C++ Producer SDK on Raspberry Pi

The Raspberry Pi is a small, inexpensive computer that can be used to teach and learn basic computer programming skills. This tutorial describes how you can set up and use the Amazon Kinesis Video Streams C++ Producer SDK on a Raspberry Pi device. The steps also include how to verify the installation using the GStreamer demo application.

Topics

Prerequisites

Before you set up the C++ Producer SDK on your Raspberry Pi, ensure that you have the following prerequisites:

- A Raspberry Pi device with the following configuration:
  - Board version: 3 Model B or later.
  - A connected camera module.
  - An SD card with a capacity of at least 8 GB.
  - The Raspbian operating system (kernel version 4.9 or later) installed. You can download the latest Raspberry Pi OS (previously called Raspbian) image from the Raspberry Pi website. Follow the Raspberry Pi instructions to install the downloaded image on an SD card.

- An AWS account with a Kinesis video stream. For more information, see Getting Started with Kinesis Video Streams.

> **Note**
>
> The C++ Producer SDK uses the US West (Oregon) (`us-west-2`) Region by default. To use the default AWS Region create your Kinesis video stream in the US West (Oregon) Region. To use a different Region for your Kinesis video stream, do one of the following:
>
> - Set the following environment variable to your Region (for example, *`us-east-1`*):
>
> ```
> export AWS_DEFAULT_REGION=us-east-1
> ```

Create an IAM user with permission to write to Kinesis Video Streams

If you haven't already done so, set up an AWS Identity and Access Management (IAM) user with permissions to write to a Kinesis video stream.

These procedures are meant to help you quickly get started using an AWS access key pair. Devices can use X.509 certificates to connect to AWS IoT. See the section called “Controlling access to Kinesis Video Streams resources using AWS IoT” for more information about how to configure your device to use certificate-based authentication.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation menu on the left, choose **Users**.
3. To create a new user, choose **Add user**.
4. Provide a descriptive **User name** for the user, such as **`kinesis-video-raspberry-pi-producer`**.
5. Under **Access type**, choose **Programmatic access**.
6. Choose **Next: Permissions**.
7. Under **Set permissions for kinesis-video-raspberry-pi-producer**, choose **Attach existing policies directly**.
8. Choose **Create policy**. The **Create policy** page opens in a new web browser tab.
9. Choose the **JSON** tab.

10. Copy the following JSON policy and paste it into the text area. This policy gives your user permission to create and write data to Kinesis video streams.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
      "Action": [
      "kinesisvideo:DescribeStream",
      "kinesisvideo:CreateStream",
      "kinesisvideo:GetDataEndpoint",
      "kinesisvideo:PutMedia"
    ],
    "Resource": [
      "*"
    ]
  }]
}
```

11. Choose **Review policy**.
12. Provide a **Name** for your policy, such as **`kinesis-video-stream-write-policy`**.
13. Choose **Create policy**.
14. Return to the **Add user** tab in your browser, and choose **Refresh**.
15. In the search box, type the name of the policy you created.
16. Select the check box next to your new policy in the list.
17. Choose **Next: Review**.
18. Choose **Create user**.
19. The console displays the **Access key ID** for your new user. Choose **Show** to display the **Secret access key**. Record these values; they are required when you configure the application.

Join your Raspberry Pi to your Wi-Fi network

You can use the Raspberry Pi in *headless* mode, that is, without an attached keyboard, monitor, or network cable. If you are using an attached monitor and keyboard, proceed to Configure the Raspberry Pi camera.

1. On your computer, create a file named `wpa_supplicant.conf`.
2. Copy the following text and paste it into the `wpa_supplicant.conf` file:

```
country=US
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
ssid="Your Wi-Fi SSID"
scan_ssid=1
key_mgmt=WPA-PSK
psk="Your Wi-Fi Password"
}
```

Replace the `ssid` and `psk` values with the information for your Wi-Fi network.

3. Copy the `wpa_supplicant.conf` file to the SD card. It must be copied to the root of the `boot` volume.
4. Insert the SD card into the Raspberry Pi, and power the device. It joins your Wi-Fi network, and SSH is enabled.

Connect remotely to your Raspberry Pi

You can connect remotely to your Raspberry Pi in headless mode. If you are using your Raspberry Pi with a connected monitor and keyboard, proceed to Configure the Raspberry Pi camera.

1. Before connecting to your Raspberry Pi device remotely, do one of the following to determine its IP address:

   - If you have access to your network's Wi-Fi router, look at the connected Wi-Fi devices. Find the device named `Raspberry Pi` to find your device's IP address.
   - If you don't have access to your network's Wi-Fi router, you can use other software to find devices on your network. Fing is a popular application that is available for both Android and iOS devices. You can use the free version of this application to find the IP addresses of devices on your network.
2. When you know the IP address of the Raspberry Pi device, you can use any terminal application to connect.

   - On macOS or Linux, use `ssh`:

   ```
   ssh pi@<IP address>
   ```

- On Windows, use PuTTY, a free SSH client for Windows.

For a new installation of Raspbian, the user name is **pi**, and the password is **raspberry**. We recommend that you change the default password.

Configure the Raspberry Pi camera

Follow these steps to configure the Raspberry Pi camera to send video from the device to a Kinesis video stream.

1. Open an editor to update the `modules` file with the following command:

   ```
   sudo nano /etc/modules
   ```

2. Add the following line to the end of the file, if it's not already there:

   ```
   bcm2835-v4l2
   ```

3. Save the file and exit the editor (Ctrl-X).
4. Reboot the Raspberry Pi:

   ```
   sudo reboot
   ```

5. When the device reboots, connect to it again through your terminal application if you are connecting remotely.
6. Open `raspi-config`:

   ```
   sudo raspi-config
   ```

7. Choose **Interfacing Options**, **Legacy Camera**. In older builds of the Raspbian Operating System, this menu option might be under **Interfacing Options**, **Camera**.

   Enable the camera if it's not already enabled, and reboot if prompted.

8. Verify that the camera is working by typing the following command:

   ```
   raspistill -v -o test.jpg
   ```

If your camera is configured correctly, this command captures an image from the camera, saves it to a file named `test.jpg`, and displays informational messages.

Install software prerequisites

The C++ Producer SDK requires that you install the following software prerequisites on Raspberry Pi.

1. Update the package list and install the libraries needed to build the SDK. Type the following commands:

```
sudo apt update
sudo apt install -y \
  automake \
  build-essential \
  cmake \
  git \
  gstreamer1.0-plugins-base-apps \
  gstreamer1.0-plugins-bad \
  gstreamer1.0-plugins-good \
  gstreamer1.0-plugins-ugly \
  gstreamer1.0-tools \
  gstreamer1.0-omx-generic \
  libcurl4-openssl-dev \
  libgstreamer1.0-dev \
  libgstreamer-plugins-base1.0-dev \
  liblog4cplus-dev \
  libssl-dev \
  pkg-config
```

2. Copy the following PEM file to `/etc/ssl/cert.pem`:

```
sudo curl https://www.amazontrust.com/repository/AmazonRootCA1.pem -o /etc/ssl/
AmazonRootCA1.pem
sudo chmod 644 /etc/ssl/AmazonRootCA1.pem
```

Download and build the Kinesis Video Streams C++ Producer SDK

You can download and build the Kinesis Video Streams C++ Producer SDK using the following procedure. This approach takes longer time to build, depending on network connectivity and processor speed.

1. Download the SDK. Type:

```
git clone https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp.git
```

2. Prepare a build directory. Type:

```
mkdir -p amazon-kinesis-video-streams-producer-sdk-cpp/build
cd amazon-kinesis-video-streams-producer-sdk-cpp/build
```

3. Build the SDK and Sample applications. Depending on which model of Raspberry Pi you are building on, it may take several hours to run for the first time:

```
cmake .. -DBUILD_GSTREAMER_PLUGIN=ON -DBUILD_DEPENDENCIES=FALSE
make
```

Stream video to your Kinesis video stream and view the live stream

1. To run the sample application, you need the following information:

   - The name of the stream you created in the Prerequisites section.
   - The account credentials (access key ID and secret access key) that you created in Create an IAM user with permission to write to Kinesis Video Streams.

2. Run the sample application using the following commands. Replace the placeholders with values for your environment.

```
export GST_PLUGIN_PATH=Directory Where You Cloned the SDK/amazon-kinesis-video-streams-producer-sdk-cpp/build
export AWS_DEFAULT_REGION=AWS Region i.e. us-east-1
export AWS_ACCESS_KEY_ID=Access Key ID
export AWS_SECRET_ACCESS_KEY=Secret Access Key
./kvs_gstreamer_sample Your Stream Name
```

3. If the sample application exits with a `library not found` error, type the following command to verify that the project is correctly linked to its open-source dependencies:

```
gst-inspect-1.0 kvssink
```

4. Open the [Kinesis Video Streams console](https://console.aws.amazon.com/kinesisvideo/).
5. Choose the **Stream name** of the stream you created.

The video stream that is sent from the Raspberry Pi appears in the console.

When the stream is playing, you can experiment with the following features of the Kinesis Video Streams console:

- In the **Video preview** section, use the navigation controls to rewind or fast-forward the stream.
- In the **Stream info** section, notice the codec, resolution, and bitrate of the stream. The resolution and bitrate values are set purposefully low on the Raspberry Pi to minimize bandwidth usage for this tutorial. To view the Amazon CloudWatch metrics that are being created for your stream, choose **View stream metrics in CloudWatch**.
- Under **Data retention period**, notice that the video stream is retained for one day. You can edit this value and set it to **No data retention**, or set a value from one day to several years.

  Under server-side encryption, notice that your data is being encrypted at rest using a key maintained by the AWS Key Management Service (AWS KMS).

Producer SDK Reference

This section contains limits, error codes, and other reference information for the Kinesis Video Streams Producer Libraries.

Topics

- Producer SDK limits
- Error code reference
- Network Abstraction Layer (NAL) adaptation flag reference
- Producer SDK structures
- Kinesis video stream structures
- Producer SDK callbacks

Producer SDK limits

The following table contains the current limits for values in the Producer Libraries.

> **Note**
>
> Before setting these values, you must validate your inputs. The SDK doesn't validate these limits, and a runtime error occurs if the limits are exceeded.

Value	Limit	Notes
Max stream count	128	The maximum number of streams that a producer object can create. This is a soft limit (you can request an increase). It guarantees that the producer doesn't accidentally create streams recursively.
Max device name length	128 characters	
Max tag count	50 per stream	
Max stream name length	256 characters	
Min storage size	10 MiB = 10 * 1024 * 1024 bytes	
Max storage size	10 GiB = 10 * 1024 * 1024 * 1024 bytes	
Max root directory path length	4,096 characters	
Max auth info length	10,000 bytes	
Max URI string length	10,000 characters	

Value	Limit	Notes
Max tag name length	128 characters	
Max tag value length	1,024 characters	
Min security token period	30 seconds	
Security token grace period	40 minutes	If the specified duration is longer, it's limited to this value.
Retention period	0 or greater than one hour	0 indicates no retention.
Min cluster duration	1 second	The value is specified in 100 ns units, which is the SDK standard.
Max cluster duration	30 seconds	The value is specified in 100 ns units, which is the SDK standard. The backend API can enforce a shorter cluster duration.
Max fragment size	50 MB	For more information, see Kinesis Video Streams service quotas.
Max fragment duration	20 seconds	For more information, see Kinesis Video Streams service quotas.
Max connection duration	45 minutes	The backend closes the connection after this time. The SDK rotates the token and establishes a new connection within this time.

Value	Limit	Notes
Max ACK segment length	1,024 characters	Maximum segment length of the acknowledgement sent to the ACK parser function.
Max content type string length	128 characters	
Max codec ID string length	32 characters	
Max track name string length	32 characters	
Max codec private data length	1 MiB = 1 * 1024 * 1024 bytes	
Min timecode scale value length	100 ns	The minimum timecode scale value to represent the frame timestamps in the resulting MKV cluster. The value is specified in increments of 100 ns, which is the SDK standard.
Max timecode scale value length	1 second	The maximum timecode scale value to represent the frame timestamps in the resulting MKV cluster. The value is specified in increments of 100 ns, which is the SDK standard.
Min content view item count	10	
Min buffer duration	20 seconds	The value is specified in increments of 100 ns, which is the SDK standard.
Max update version length	128 characters	
Max ARN length	1024 characters	

Value	Limit	Notes
Max fragment sequence length	128 characters	
Max retention period	10 years	

Error code reference

This section contains error and status code information for the Producer Libraries.

For information about solutions to common issues, see Troubleshooting Kinesis Video Streams.

Topics

- Errors and status codes returned by PutFrame Callbacks - Platform Independent Code (PIC)
- Errors and status codes returned by PutFrame callbacks - C producer library

Errors and status codes returned by PutFrame Callbacks - Platform Independent Code (PIC)

The following sections contain error and status information that are returned by callbacks for the `PutFrame` operation within the Platform Independent Code (PIC).

Topics

- Error and status codes returned by the client library
- Error and status codes returned by the duration library
- Error and status codes returned by the common library
- Error and status codes returned by the heap library
- Error and status codes returned by the MKVGen library
- Error and status codes returned by the Trace library
- Error and status codes returned by the Utils library
- Error and status codes returned by the View library

Error and status codes returned by the client library

The following table contains error and status information that are returned by methods in the Kinesis Video Streams `Client` library.

Code	Message	Description	Recommended action
`0x52000001`	`STATUS_MAX_STREAM_COUNT`	The maximum stream count was reached.	Specify a larger max stream count in `DeviceInfo` as specified in Producer SDK limits.
`0x52000002`	`STATUS_MIN_STREAM_COUNT`	Minimum stream count error.	Specify the max number of streams greater than zero in `DeviceInfo` .
`0x52000003`	`STATUS_INVALID_DEVICE_NAME_LENGTH`	Invalid device name length.	Refer to the max device name length in characters that's specified in Producer SDK limits.
`0x52000004`	`STATUS_INVALID_DEVICE_INFO_VERSION`	Invalid `DeviceInfo` structure version.	Specify the correct current version of the structure.
`0x52000005`	`STATUS_MAX_TAG_COUNT`	The maximum tag count was reached.	Refer to the current max tag count that's specified in Producer SDK limits.
`0x52000006`	`STATUS_DEVICE_FINGERPRINT_LENGTH`		
`0x52000007`	`STATUS_INVALID_CALLBACKS_VERSION`	Invalid `Callbacks` structure version.	Specify the correct current version of the structure.

Code	Message	Description	Recommended action
0x52000008	STATUS_INVALID_STREAM_INFO_VERSION	Invalid StreamInfo structure version.	Specify the correct current version of the structure.
0x52000009	STATUS_INVALID_STREAM_NAME_LENGTH	Invalid stream name length.	Refer to the max stream name length in characters that's specified in Producer SDK limits.
0x5200000a	STATUS_INVALID_STORAGE_SIZE	An invalid storage size was specified.	The storage size in bytes must be within the limits specified in Producer SDK limits.
0x5200000b	STATUS_INVALID_ROOT_DIRECTORY_LENGTH	Invalid root directory string length.	Refer to the max root directory path length that's specified in Producer SDK limits.
0x5200000c	STATUS_INVALID_SPILL_RATIO	Invalid spill ratio.	Express the spill ratio as a percentage from 0–100.
0x5200000d	STATUS_INVALID_STORAGE_INFO_VERSION	Invalid StorageInfo structure version.	Specify the correct current version of the structure.

Code	Message	Description	Recommended action
0x5200000e	STATUS_INVALID_STREAM_STATE	The stream is in a state that doesn't permit the current operation.	Most commonly, this error occurs when the SDK fails to reach the state that it requires to perform the requested operation. For example, it occurs if the `GetStreamingEndpoint` API call fails, and the client application ignores it and continues putting frames into the stream.
0x5200000f	STATUS_SERVICE_CALL_CALLBACKS_MISSING	The `Callbacks` structure has missing function entry points for some mandatory functions.	Verify that the mandatory callbacks are implemented in the client application. This error is exposed only to Platform Independent Code (PIC) clients. C++ and other higher-level wrappers satisfy these calls.
0x52000010	STATUS_SERVICE_CALL_NOT_AUTHORIZED_ERROR	Not authorized.	Verify the security token, certificate, security token integration, and expiration. Verify that the token has the correct associated rights with it. For the Kinesis Video Streams sample applications, verify that the environment variable is set correctly.

Code	Message	Description	Recommended action
0x52000011	STATUS_DESCRIBE_STREAM_CALL_FAILED	DescribeStream API failure.	This error is returned after the DescribeStream API retry failure. The PIC client returns this error after it stops retrying.
0x52000012	STATUS_INVALID_DESCRIBE_STREAM_RESPONSE	Invalid DescribeStreamResponse structure.	The structure that was passed to the DescribeStreamResultEvent is either null or contains invalid items like a null Amazon Resource Name (ARN).
0x52000013	STATUS_STREAM_IS_BEING_DELETED_ERROR	The stream is being deleted.	An API failure was caused by the stream being deleted. Verify that no other processes are trying to delete the stream while the stream is in use.
0x52000014	STATUS_SERVICE_CALL_INVALID_ARG_ERROR	Invalid arguments were specified for the service call.	The backend returns this error when a service call argument isn't valid or when the SDK encounters an error that it can't interpret.
0x52000015	STATUS_SERVICE_CALL_DEVICE_NOT_FOUND_ERROR	The device was not found.	Verify that the device isn't deleted while in use.

Code	Message	Description	Recommended action
0x52000016	STATUS_SERVICE_CALL_DEVICE_NOT_PROVISIONED_ERROR	The device was not provisioned.	Verify that the device has been provisioned.
0x52000017	STATUS_SERVICE_CALL_RESOURCE_NOT_FOUND_ERROR	Generic resource not found returned from the service.	This error occurs when the service can't locate the resource (for example, a stream). It might mean different things in different contexts, but the likely cause is the usage of APIs before the stream is created. Using the SDK confirms that the stream is created first.
0x52000018	STATUS_INVALID_AUTH_LEN	Invalid auth info length.	Refer to the current values that are specified in [Producer SDK limits]().
0x52000019	STATUS_CREATE_STREAM_CALL_FAILED	The `CreateStream` API call failed.	Refer to the error string for more detailed information about why the operation failed.
0x5200002a	STATUS_GET_STREAMING_TOKEN_CALL_FAILED	The `GetStreamingToken` call failed.	Refer to the error string for more detailed information about why the operation failed.

Code	Message	Description	Recommended action
0x5200002b	STATUS_GET_STREAMING_ENDPOINT_CALL_FAILED	The GetStreamingEndpoint API call failed.	Refer to the error string for more detailed information about why the operation failed.
0x5200002c	STATUS_INVALID_URI_LEN	An invalid URI string length was returned from the GetStreamingEndpoint API.	Refer to the current maximum values that are specified in Producer SDK limits.
0x5200002d	STATUS_PUT_STREAM_CALL_FAILED	The PutMedia API call failed.	Refer to the error string for more detailed information about why the operation failed.

Code	Message	Description	Recommended action
0x5200002e	STATUS_STORE_OUT_OF_MEMORY	The content store is out of memory.	The content store is shared between the streams and should have enough capacity to store the maximum durations for all the streams + ~20% (accounting for the defragmentation). It's important to not overflow the storage. Choose values for the maximum duration per stream that correspond to the cumulative storage size and the latency tolerances. We recommend dropping the frames as they fall out of the content view window versus just being put (content store memory pressure). This is because dropping the frames initiates the stream pressure notification callbacks. Then the application can adjust the upstream media components (like the encoder) to thin the bitrate, drop frames, or act accordingly.

Code	Message	Description	Recommended action
0x5200002f	STATUS_NO_MORE_DATA_AVAILABLE	No more data is available currently for a stream.	This is a potential valid result when the media pipeline produces more slowly than the networking thread consumes the frames to be sent to the service. Higher-level clients (for example, C++, Java, or Android) don't see this warning because it's handled internally.
0x52000030	STATUS_INVALID_TAG_VERSION	Invalid Tag structure version.	Specify the correct current version of the structure.
0x52000031	STATUS_SERVICE_CALL_UNKNOWN_ERROR	An unknown or generic error was returned from the networking stack.	See the logs for more detailed information.
0x52000032	STATUS_SERVICE_CALL_RESOURCE_IN_USE_ERROR	Resource in use.	Returned from the service. For more information, see the Kinesis Video Streams API Reference.
0x52000033	STATUS_SERVICE_CALL_CLIENT_LIMIT_ERROR	Client limit.	Returned from the service. For more information, see the Kinesis Video Streams API Reference.

Code	Message	Description	Recommended action
0x52000034	STATUS_SERVICE_CALL_DEVICE_LIMIT_ERROR	Device limit.	Returned from the service. For more information, see the Kinesis Video Streams API Reference.
0x52000035	STATUS_SERVICE_CALL_STREAM_LIMIT_ERROR	Stream limit.	Returned from the service. For more information, see the Kinesis Video Streams API Reference.
0x52000036	STATUS_SERVICE_CALL_RESOURCE_DELETED_ERROR	The resource was deleted or is being deleted.	Returned from the service. For more information, see the Kinesis Video Streams API Reference.
0x52000037	STATUS_SERVICE_CALL_TIMEOUT_ERROR	The service call timed out.	Calling a particular service API resulted in a timeout. Verify that you have a valid network connection. The PIC will retry the operation automatically.
0x52000038	STATUS_STREAM_READY_CALLBACK_FAILED	Stream ready notification.	This notification is sent from the PIC to the client indicating that the async stream has been created.
0x52000039	STATUS_DEVICE_TAGS_COUNT_NON_ZERO_TAGS_NULL	Invalid tags were specified.	The tag count isn't zero, but the tags are empty. Verify that the tags are specified or the count is zero.

Code	Message	Description	Recommended action
0x5200003a	STATUS_INVALID_STREAM_DESCRIPTION_VERSION	Invalid StreamDescription structure version.	Specify the correct current version of the structure.
0x5200003b	STATUS_INVALID_TAG_NAME_LEN	Invalid tag name length.	Refer to the limits for the tag name that are specified in Producer SDK limits.
0x5200003c	STATUS_INVALID_TAG_VALUE_LEN	Invalid tag value length.	Refer to the limits for the tag value that are specified in Producer SDK limits.
0x5200003d	STATUS_TAG_STREAM_CALL_FAILED	The TagResource API failed.	The TagResource API call failed. Check for a valid network connection. See the logs for more information about the failure.
0x5200003e	STATUS_INVALID_CUSTOM_DATA	Invalid custom data calling PIC APIs.	Invalid custom data has been specified in a call to the PIC APIs. This can occur only in the clients that directly use PIC.
0x5200003f	STATUS_INVALID_CREATE_STREAM_RESPONSE	Invalid CreateStreamResponse structure.	The structure or its member fields are invalid (that is, the ARN is null or larger than what's specified in Producer SDK limits).

Code	Message	Description	Recommended action
0x52000040	STATUS_CLIENT_AUTH_CALL_FAILED	Client auth failed.	The PIC failed to get proper auth information (AccessKeyId or SecretAccessKey) after a number of retries. Check the authentication integration. The sample applications use environment variables to pass in credential information to the C++ Producer Library.
0x52000041	STATUS_GET_CLIENT_TOKEN_CALL_FAILED	Getting the security token call failed.	This situation can occur for clients that use PIC directly. After a number of retries, the call fails with this error.
0x52000042	STATUS_CLIENT_PROVISION_CALL_FAILED	Provisioning error.	Provisioning isn't implemented.
0x52000043	STATUS_CREATE_CLIENT_CALL_FAILED	Failed to create the producer client.	A generic error returned by the PIC after a number of retries when the client creation fails.
0x52000044	STATUS_CLIENT_READY_CALLBACK_FAILED	Failed to get the producer client to a READY state.	Returned by the PIC state machine if the PIC fails to move to the READY state. See the logs for more information about the root cause.

Code	Message	Description	Recommended action
0x52000045	STATUS_TAG_CLIENT_CALL_FAILED	The TagResource for the producer client failed.	The TagResource API call failed for the producer client. See the logs for more information about the root cause.
0x52000046	STATUS_INVALID_CREATE_DEVICE_RESPONSE	Device/Producer creation failed.	The higher-level SDKs (for example, C++ or Java) don't implement the device or producer creation API yet. Clients that use PIC directly can indicate a failure using the result notification.
0x52000047	STATUS_ACK_TIMESTAMP_NOT_IN_VIEW_WINDOW	The timestamp of the received ACK is not in the view.	This error occurs if the frame corresponding to the received ACK falls out of the content view window. Generally, this occurs if the ACK delivery is slow. It can be interpreted as a warning and an indication that the downlink is slow.
0x52000048	STATUS_INVALID_FRAGMENT_ACK_VERSION	Invalid FragmentAck structure version.	Specify the correct current version of the FragmentAck structure.

Code	Message	Description	Recommended action
`0x52000049`	`STATUS_INVALID_TOKEN_EXPIRATION`	Invalid security token expiration.	The security token expiration should have an absolute timestamp in the future that's greater than the current timestamp, with a grace period. For the limits for the grace period, see the Producer SDK limits.
`0x5200004a`	`STATUS_END_OF_STREAM`	End of stream (EOS) indicator.	In the `GetStreamData` API call, indicates that the current upload handle session has ended. This occurs if the session ends or errors, or if the session token has expired and the session is being rotated.
`0x5200004b`	`STATUS_DUPLICATE_STREAM_NAME`	Duplicate stream name.	Multiple streams can't have the same stream name. Choose a unique name for the stream.
`0x5200004c`	`STATUS_INVALID_RETENTION_PERIOD`	Invalid retention period.	An invalid retention period is specified in the `StreamInfo` structure. For information about the valid range of values for the retention period, see Producer SDK limits.

Code	Message	Description	Recommended action
0x5200004d	STATUS_INVALID_ACK_KEY_START	Invalid FragmentAck .	Failed to parse the fragment ACK string. Invalid key start indicator. The fragment ACK string might be damaged. It can self-correct and this error can be treated as a warning.
0x5200004e	STATUS_INVALID_ACK_DUPLICATE_KEY_NAME	Invalid FragmentAck .	Failed to parse the fragment ACK string. Multiple keys have the same name. The fragment ACK string might be damaged. It can self-correct and this error can be treated as a warning.
0x5200004f	STATUS_INVALID_ACK_INVALID_VALUE_START	Invalid FragmentAck .	Failed to parse the fragment ACK string because of an invalid key value start indicator. The fragment ACK string might be damaged. It can self-correct, and this error can be treated as a warning.

Code	Message	Description	Recommended action
0x52000050	STATUS_INVALID_ACK_INVALID_VALUE_END	Invalid FragmentAck .	Failed to parse the fragment ACK string because of an invalid key value end indicator. The fragment ACK string might be damaged. It can self-correct and this error can be treated as a warning.
0x52000051	STATUS_INVALID_PARSED_ACK_TYPE	Invalid FragmentAck .	Failed to parse the fragment ACK string because an invalid ACK type was specified.
0x52000052	STATUS_STREAM_HAS_BEEN_STOPPED	Stream was stopped.	The stream has been stopped, but a frame is still being put into the stream.
0x52000053	STATUS_INVALID_STREAM_METRICS_VERSION	Invalid StreamMetrics structure version.	Specify the correct current version of the StreamMetrics structure.
0x52000054	STATUS_INVALID_CLIENT_METRICS_VERSION	Invalid ClientMetrics structure version.	Specify the correct current version of the ClientMetrics structure.
0x52000055	STATUS_INVALID_CLIENT_READY_STATE	Producer initialization failed to reach a READY state.	Failed to reach the READY state during the producer client initialization. See the logs for more information.

Code	Message	Description	Recommended action
`0x52000056`	`STATUS_STATE_MACHINE_STATE_NOT_FOUND`	Internal state machine error.	Not a publicly visible error.
`0x52000057`	`STATUS_INVALID_FRAGMENT_ACK_TYPE`	Invalid ACK type is specified in the `FragmentAck` structure.	The `FragmentAck` structure should contain ACK types defined in the public header.
`0x52000058`	`STATUS_INVALID_STREAM_READY_STATE`	Internal state machine transition error.	Not a publicly visible error.
`0x52000059`	`STATUS_CLIENT_FREED_BEFORE_STREAM`	The stream object was freed after the producer was freed.	There was an attempt to free a stream object after the producer object was freed. This can only occur in clients that directly use PIC.
`0x5200005a`	`STATUS_ALLOCATION_SIZE_SMALLER_THAN_REQUESTED`	Internal storage error.	An internal error indicating that the actual allocation size from the content store is smaller than the size of the packaged frame and fragment.
`0x5200005b`	`STATUS_VIEW_ITEM_SIZE_GREATER_THAN_ALLOCATION`	Internal storage error.	The stored size of the allocation in the content view is greater than the allocation size in the content store.

Code	Message	Description	Recommended action
0x5200005c	STATUS_ACK_ERR_STREAM_READ_ERROR	Stream read error ACK.	An error that the ACK returned from the backend indicating a stream read or parsing error. This generally occurs when the backend fails to retrieve the stream. Auto-restreaming can usually correct this error.
0x5200005d	STATUS_ACK_ERR_FRAGMENT_SIZE_REACHED	The maximum fragment size was reached.	The max fragment size in bytes is defined in Producer SDK limits. This error indicates that there are either very large frames, or there are no key frames to create manageable size fragments. Check the encoder settings and verify that key frames are being produced properly. For streams that have very high density, configure the encoder to produce fragments at smaller durations to manage the maximum size.

Code	Message	Description	Recommended action
0x5200005e	STATUS_ACK_ERR_FRAGMENT_DURATION_REACHED	The maximum fragment duration was reached.	The max fragment duration is defined in [Producer SDK limits](). This error indicates that there are either very low frames per second or there are no key frames to create manageable duration fragments. Check the encoder settings and verify that key frames are being produced properly at the regular intervals.
0x5200005f	STATUS_ACK_ERR_CONNECTION_DURATION_REACHED	The maximum connection duration was reached.	Kinesis Video Streams enforces the max connection duration as specified in the [Producer SDK limits](). The Producer SDK automatically rotates the stream or token before the maximum is reached. Clients using the SDK shouldn't receive this error.
0x52000060	STATUS_ACK_ERR_FRAGMENT_TIMECODE_NOT_MONOTONIC	Timecodes are not monotonically increasing.	The Producer SDK enforces timestamps, so clients using the SDK shouldn't receive this error.

Code	Message	Description	Recommended action
0x52000061	STATUS_ACK_ERR_MULTI_TRACK_MKV	Multiple tracks in the MKV.	The Producer SDK enforces single track streams, so clients using the SDK shouldn't receive this error.
0x52000062	STATUS_ACK_ERR_INVALID_MKV_DATA	Invalid MKV data.	The backend MKV parser encountered an error parsing the stream. Clients using the SDK might encounter this error if the stream is corrupted in the transition. This can also happen if the buffer pressures force the SDK to drop tail frames that are partially transmitted. In the latter case, we recommend that you either reduce the FPS and resolution, increase the compression ratio, or (if there's a "bursty" network) allow for larger content store and buffer duration to accommodate for the temporary pressures.

Code	Message	Description	Recommended action
0x52000063	STATUS_ACK_ERR_INVALID_PRODUCER_TIMESTAMP	Invalid producer timestamp.	The service returns this error ACK if the producer clock has a large drift into the future. Higher-level SDKs (for example, Java or C++) use some version of the system clock to satisfy the current time callback from PIC. Verify that the system clock is set properly. Clients using the PIC directly should verify that their callback functions return the correct timestamp.
0x52000064	STATUS_ACK_ERR_STREAM_NOT_ACTIVE	Inactive stream.	A call to a backend API was made while the stream was not in an "Active" state. This occurs when the client creates the stream and immediately continues to push frames into it. The SDK handles this scenario through the state machine and recovery mechanism.
0x52000065	STATUS_ACK_ERR_KMS_KEY_ACCESS_DENIED	AWS KMS access denied error.	Returned when the account has no access to the specified key.

Code	Message	Description	Recommended action
0x52000066	STATUS_ACK_ERR_KMS_KEY_DISABLED	AWS KMS key is disabled.	The specified key has been disabled.
0x52000067	STATUS_ACK_ERR_KMS_KEY_VALIDATION_ERROR	AWS KMS key validation error.	Generic validation error. For more information, see the *AWS Key Management Service API Reference*.
0x52000068	STATUS_ACK_ERR_KMS_KEY_UNAVAILABLE	AWS KMS key unavailable.	The key is unavailable. For more information, see the *AWS Key Management Service API Reference*.
0x52000069	STATUS_ACK_ERR_KMS_KEY_INVALID_USAGE	Invalid use of KMS key.	The AWS KMS key is not configured to be used in this context. For more information, see the *AWS Key Management Service API Reference*.
0x5200006a	STATUS_ACK_ERR_KMS_KEY_INVALID_STATE	AWS KMS invalid state.	For more information, see the *AWS Key Management Service API Reference*.
0x5200006b	STATUS_ACK_ERR_KMS_KEY_NOT_FOUND	KMS key not found.	The key was not found. For more information, see the *AWS Key Management Service API Reference*.
0x5200006c	STATUS_ACK_ERR_STREAM_DELETED	The stream has been or is being deleted.	The stream is being deleted by another application or through the AWS Management Console.

Code	Message	Description	Recommended action
0x5200006d	STATUS_ACK_ERR_ACK_INTERNAL_ERROR	Internal error.	Generic service internal error.
0x5200006e	STATUS_ACK_ERR_FRAGMENT_ARCHIVAL_ERROR	Fragment archival error.	Returned when the service fails to durably persist and index the fragment. Although it's rare, it can occur for various reasons. By default, the SDK retries sending the fragment.
0x5200006f	STATUS_ACK_ERR_UNKNOWN_ACK_ERROR	Unknown error.	The service returned an unknown error.
0x52000070	STATUS_MISSING_ERR_ACK_ID	Missing ACK information.	The ACK parser completed parsing, but the `FragmentAck` information is missing.
0x52000071	STATUS_INVALID_ACK_SEGMENT_LEN	Invalid ACK segment length.	An ACK segment string with an invalid length was specified to the ACK parser. For more information, see Producer SDK limits.

Code	Message	Description	Recommended action
0x52000074	STATUS_MAX_FRAGMENT_METADATA_COUNT	The maximum number of metadata items has been added to a fragment.	A Kinesis video stream can add up to 10 metadata items to a fragment, either by adding a nonpersistent item to a fragment, or by adding a persistent item to the metadata queue. For more information, see Using streaming metadata with Kinesis Video Streams.
0x52000075	STATUS_ACK_ERR_FRAGMENT_METADATA_LIMIT_REACHED	A limit (maximum metadata count, metadata name length, or metadata value length) has been reached.	The Producer SDK limits the number and size of metadata items. This error doesn't occur unless the limits in the Producer SDK code are changed. For more information, see Using streaming metadata with Kinesis Video Streams.
0x52000076	STATUS_BLOCKING_PUT_INTERRUPTED_STREAM_TERMINATED	Not implemented.	

Code	Message	Description	Recommended action
0x52000077	STATUS_INVALID_METADATA_NAME	The metadata name is not valid.	The metadata name can't start with the string "AWS". If this error occurs, the metadata item isn't added to the fragment or metadata queue. For more information, see Using streaming metadata with Kinesis Video Streams.
0x52000078	STATUS_END_OF_FRAGMENT_FRAME_INVALID_STATE	The end of a fragment frame is in an invalid state.	The end of fragment shouldn't be sent in a non-key-frame fragmented stream.
0x52000079	STATUS_TRACK_INFO_MISSING	Track information is missing.	The track number must be greater than zero and match the track id.
0x5200007a	STATUS_MAX_TRACK_COUNT_EXCEEDED	Maximum track count is exceeded.	You can have a maximum of three tracks per stream.
0x5200007b	STATUS_OFFLINE_MODE_WITH_ZERO_RETENTION	The offline streaming mode retention time is set to zero.	The offline streaming mode retention time shouldn't be set to zero.
0x5200007c	STATUS_ACK_ERR_TRACK_NUMBER_MISMATCH	The track number of the error ACK is mismatched.	

Code	Message	Description	Recommended action
0x5200007d	STATUS_ACK_ERR_FRAMES_MISSING_FOR_TRACK	Frames missing for a track.	
0x5200007e	STATUS_ACK_ERR_MORE_THAN_ALLOWED_TRACKS_FOUND	Maximum allowed number of tracks is exceeded.	
0x5200007f	STATUS_UPLOAD_HANDLE_ABORTED	Upload handle is aborted.	
0x52000080	STATUS_INVALID_CERT_PATH_LENGTH	Invalid certificate path length.	
0x52000081	STATUS_DUPLICATE_TRACK_ID_FOUND	Duplicate track ID found.	
0x52000082	STATUS_INVALID_CLIENT_INFO_VERSION		
0x52000083	STATUS_INVALID_CLIENT_ID_STRING_LENGTH		

Code	Message	Description	Recommended action
0x52000084	STATUS_SETTING_KEY_FRAME_FLAG_WHILE_USING_EOFR		
0x52000085	STATUS_MAX_FRAME_TIMESTAMP_DELTA_BETWEEN_TRACKS_EXCEEDED		
0x52000086	STATUS_STREAM_SHUTTING_DOWN		
0x52000087	STATUS_CLIENT_SHUTTING_DOWN		
0x52000088	STATUS_PUTMEDIA_LAST_PERSIST_ACK_NOT_RECEIVED		
0x52000089	STATUS_NON_ALIGNED_HEAP_WITH_IN_CONTENT_STORE_ALLOCATORS		

Code	Message	Description	Recommended action
0x5200008a	STATUS_MULTIPLE_CONSECUTIVE_EOFR		
0x5200008b	STATUS_DUPLICATE_STREAM_EVENT_TYPE		
0x5200008c	STATUS_STREAM_NOT_STARTED		
0x5200008d	STATUS_INVALID_IMAGE_PREFIX_LENGTH		
0x5200008e	STATUS_INVALID_IMAGE_METADATA_KEY_LENGTH		
0x5200008f	STATUS_INVALID_IMAGE_METADATA_VALUE_LENGTH		

Error and status codes returned by the duration library

The following table contains error and status information that are returned by methods in the `Duration` library.

Code	Message
0xFFFFFFFFFFFFFFFF	INVALID_DURATION_VALUE

Error and status codes returned by the common library

The following table contains error and status information that are returned by methods in the Common library.

Note

These error and status information codes are common to many APIs.

Code	Code without leading 0s	Message	Description
0x00000001	0x1	STATUS_NULL_ARG	NULL was passed for a mandatory argument.
0x00000002	0x2	STATUS_INVALID_ARG	An invalid value was specified for an argument.
0x00000003	0x3	STATUS_INVALID_ARG_LEN	An invalid argument length was specified.
0x00000004	0x4	STATUS_NOT_ENOUGH_MEMORY	Couldn't allocate enough memory.
0x00000005	0x5	STATUS_BUFFER_TOO_SMALL	The specified buffer size is too small.
0x00000006	0x6	STATUS_UNEXPECTED_EOF	An unexpected end of file was reached.
0x00000007	0x7	STATUS_FORMAT_ERROR	An invalid format was encountered.

Code	Code without leading 0s	Message	Description
0x00000008	0x8	STATUS_INVALID_HANDLE_ERROR	Invalid handle value.
0x00000009	0x9	STATUS_OPEN_FILE_FAILED	Failed to open a file.
0x0000000a	0xa	STATUS_READ_FILE_FAILED	Failed to read from a file.
0x0000000b	0xb	STATUS_WRITE_TO_FILE_FAILED	Failed to write to a file.
0x0000000c	0xc	STATUS_INTERNAL_ERROR	An internal error that normally doesn't occur and might indicate an SDK or service API bug.
0x0000000d	0xd	STATUS_INVALID_OPERATION	There was an invalid operation, or the operation is not permitted.
0x0000000e	0xe	STATUS_NOT_IMPLEMENTED	The feature is not implemented.
0x0000000f	0xf	STATUS_OPERATION_TIMED_OUT	The operation timed out.
0x00000010	0x10	STATUS_NOT_FOUND	A required resource was not found.

Code	Code without leading 0s	Message	Description
0x00000011	0x11	STATUS_CREATE_THREAD_FAILED	Failed to create a thread.
0x00000012	0x12	STATUS_THREAD_NOT_ENOUGH_RESOURCES	Insufficient resources to create another thread, or a system-imposed limit on the number of threads was encountered.
0x00000013	0x13	STATUS_THREAD_INVALID_ARG	Invalid thread attributes specified , or another thread is already waiting to join with this thread.
0x00000014	0x14	STATUS_THREAD_PERMISSIONS	No permission to set the scheduling policy and parameters specified in thread attributes.
0x00000015	0x15	STATUS_THREAD_DEADLOCKED	A deadlock was detected or the joining thread specifies the calling thread.
0x00000016	0x16	STATUS_THREAD_DOES_NOT_EXIST	Unable to find the thread with the specified thread ID.

Code	Code without leading 0s	Message	Description
0x00000017	0x17	STATUS_JOIN_THREAD_FAILED	An unknown or generic error was returned from the thread join operation.
0x00000018	0x18	STATUS_WAIT_FAILED	Exceeded the maximum time to wait for the conditional variable.
0x00000019	0x19	STATUS_CANCEL_THREAD_FAILED	An unknown or generic error was returned from the thread cancel operation.
0x0000001a	0x1a	STATUS_THREAD_IS_NOT_JOINABLE	The thread join operation is requested on a non-joinable thread.
0x0000001b	0x1b	STATUS_DETACH_THREAD_FAILED	An unknown or generic error was returned from the thread detach operation.
0x0000001c	0x1c	STATUS_THREAD_ATTR_INIT_FAILED	Failed to initialize the thread attributes object.
0x0000001d	0x1d	STATUS_THREAD_ATTR_SET_STACK_SIZE_FAILED	Failed to set the stack size for the thread attributes object.

Code	Code without leading 0s	Message	Description
0x0000001e	0x1e	STATUS_MEMORY_NOT_FREED	Only used in the tests. Indicates that not all requested memory has been freed.

Error and status codes returned by the heap library

The following table contains error and status information that are returned by methods in the Heap library.

Code	Message	Description
0x10000001	STATUS_HEAP_FLAGS_ERROR	An invalid combination of flags was specified.
0x10000002	STATUS_HEAP_NOT_INITIALIZED	An operation was attempted before the heap was initialized.
0x10000003	STATUS_HEAP_CORRUPTED	The heap was corrupted or the guard band (in debug mode) was overwritten. A buffer overflow in the client code might lead to a heap corruption.
0x10000004	STATUS_HEAP_VRAM_LIB_MISSING	The VRAM (video RAM) user or kernel mode library can't be loaded or is missing. Check if the underlying platform supports VRAM allocations.

Code	Message	Description
0x10000005	STATUS_HEAP_VRAM_LIB_REOPEN	Failed to open the VRAM library.
0x10000006	STATUS_HEAP_VRAM_INIT_FUNC_SYMBOL	Failed to load the INIT function export.
0x10000007	STATUS_HEAP_VRAM_ALLOC_FUNC_SYMBOL	Failed to load the ALLOC function export.
0x10000008	STATUS_HEAP_VRAM_FREE_FUNC_SYMBOL	Failed to load the FREE function export.
0x10000009	STATUS_HEAP_VRAM_LOCK_FUNC_SYMBOL	Failed to load the LOCK function export.
0x1000000a	STATUS_HEAP_VRAM_UNLOCK_FUNC_SYMBOL	Failed to load the UNLOCK function export.
0x1000000b	STATUS_HEAP_VRAM_UNINIT_FUNC_SYMBOL	Failed to load the UNINIT function export.
0x1000000c	STATUS_HEAP_VRAM_GETMAX_FUNC_SYMBOL	Failed to load the GETMAX function export.
0x1000000d	STATUS_HEAP_DIRECT_MEM_INIT	Failed to initialize the main heap pool in the hybrid heap.
0x1000000e	STATUS_HEAP_VRAM_INIT_FAILED	The VRAM dynamic initialization failed.
0x1000000f	STATUS_HEAP_LIBRARY_FREE_FAILED	Failed to de-allocate and free the VRAM library.
0x10000010	STATUS_HEAP_VRAM_ALLOC_FAILED	The VRAM allocation failed.

Code	Message	Description
0x10000011	STATUS_HEAP_VRAM_FREE_FAILED	The VRAM free failed.
0x10000012	STATUS_HEAP_VRAM_MAP_FAILED	The VRAM map failed.
0x10000013	STATUS_HEAP_VRAM_UNMAP_FAILED	The VRAM unmap failed.
0x10000014	STATUS_HEAP_VRAM_UNINIT_FAILED	The VRAM deinitialization failed.
0x10000015	STATUS_INVALID_ALLOCATION_SIZE	
0x10000016	STATUS_HEAP_REALLOC_ERROR	
0x10000017	STATUS_HEAP_FILE_HEAP_FILE_CORRUPT	

Error and status codes returned by the MKVGen library

The following table contains error and status information that are returned by methods in the `MKVGen` library.

Code	Message	Description / Recommended action
0x32000001	STATUS_MKV_INVALID_FRAME_DATA	Invalid members of the `Frame` data structure. Make sure that the duration, size, and frame data are valid and are within the limits specified in [Producer SDK limits](#).

Code	Message	Description / Recommended action
0x32000002	STATUS_MKV_INVALID_FRAME_TIMESTAMP	Invalid frame timestamp. The calculated PTS (presentation timestamp) and DTS (decoding timestamp) are greater or equal to the timestamp of the start frame of the fragment. This is an indication of a potential media pipeline restart or an encoder stability issue. For troubleshooting information, see Error: "Failed to submit frame to Kinesis Video client"
0x32000003	STATUS_MKV_INVALID_CLUSTER_DURATION	An invalid fragment duration was specified. For more information, see Producer SDK limits.
0x32000004	STATUS_MKV_INVALID_CONTENT_TYPE_LENGTH	Invalid content type string length. For more information, see Producer SDK limits.
0x32000005	STATUS_MKV_NUMBER_TOO_BIG	There was an attempt to encode a number that's too large to be represented in EBML (Extensible Binary Meta Language) format. This should not be exposed to the SDK clients.
0x32000006	STATUS_MKV_INVALID_CODEC_ID_LENGTH	Invalid codec ID string length. For more information, see Producer SDK limits.

Code	Message	Description / Recommended action
0x32000007	STATUS_MKV_INVALID_TRACK_NAME_LENGTH	Invalid track name string length. For more information, see Producer SDK limits.
0x32000008	STATUS_MKV_INVALID_CODEC_PRIVATE_LENGTH	Invalid codec private data length. For more information, see Producer SDK limits.
0x32000009	STATUS_MKV_CODEC_PRIVATE_NULL	The codec private data (CPD) is NULL, whereas the CPD size is greater than zero.
0x3200000a	STATUS_MKV_INVALID_TIMECODE_SCALE	Invalid timecode scale value. For more information, see Producer SDK limits.
0x3200000b	STATUS_MKV_MAX_FRAME_TIMECODE	The frame timecode is greater than the maximum. For more information, see Producer SDK limits.

Code	Message	Description / Recommended action
0x3200000c	STATUS_MKV_LARGE_FRAME_TIMECODE	The max frame timecode was reached. The MKV format uses signed 16 bits to represent the relative timecode of the frame to the beginning of the cluster. The error is generated if the frame timecode can't be represented. This error indicates either a bad timecode scale selection or the cluster duration is too long, so representing the frame timecode overflows the signed 16-bit space.

Code	Message	Description / Recommended action
0x3200000d	STATUS_MKV_INVALID_ANNEXB_NALU_IN_FRAME_DATA	An invalid Annex-B start code was encountered. For example, the Annex-B adaptation flag was specified and the code encounters an invalid start sequence of more than three zeroes. A valid Annex-B format should have an "emulation prevention" sequence to escape a sequence of three or more zeroes in the bytestream. For more information, see the MPEG specification. For information about this error on Android, see STATUS_MKV_INVALID_ANNEXB_NALU_IN_FRAME_DATA (0x3200000d) error on Android.
0x3200000e	STATUS_MKV_INVALID_AVCC_NALU_IN_FRAME_DATA	Invalid AVCC NALU packaging when the adapting AVCC flag is specified. Verify that the bytestream is in a valid AVCC format. For more information, see the MPEG specification.
0x3200000f	STATUS_MKV_BOTH_ANNEXB_AND_AVCC_SPECIFIED	Both adapting AVCC and Annex-B NALUs were specified. Specify either one, or specify none.

Code	Message	Description / Recommended action
0x32000010	STATUS_MKV_INVALID_ANNEXB_NALU_IN_CPD	Invalid Annex-B format of CPD when the adapting Annex-B flag is specified. Verify that the CPD is in valid Annex-B format. If it's not, then remove the CPD Annex-B adaptation flag.
0x32000011	STATUS_MKV_PTS_DTS_ARE_NOT_SAME	Kinesis Video Streams enforces the PTS (presentation timestamp) and DTS (decoding timestamp) to be the same for the fragment start frames. These are the key frames that start the fragment.
0x32000012	STATUS_MKV_INVALID_H264_H265_CPD	Failed to parse H264/H265 codec private data.
0x32000013	STATUS_MKV_INVALID_H264_H265_SPS_WIDTH	Failed to extract the width from the codec private data.
0x32000014	STATUS_MKV_INVALID_H264_H265_SPS_HEIGHT	Failed to extract the height from codec private data.
0x32000015	STATUS_MKV_INVALID_H264_H265_SPS_NALU	Invalid H264/H265 SPS NALU.
0x32000016	STATUS_MKV_INVALID_BIH_CPD	Invalid bitmap info header format in the codec private data.

Code	Message	Description / Recommended action
0x32000017	STATUS_MKV_INVALID_HEVC_NALU_COUNT	Invalid High Efficiency Video Coding (HEVC) Network Abstraction Layer units (NALU) count.
0x32000018	STATUS_MKV_INVALID_HEVC_FORMAT	Invalid HEVC format.
0x32000019	STATUS_MKV_HEVC_SPS_NALU_MISSING	Missing HEVC NALUs in the Sequence Parameter Set (SPS).
0x3200001a	STATUS_MKV_INVALID_HEVC_SPS_NALU_SIZE	Invalid HEVC SPS NALU size.
0x3200001b	STATUS_MKV_INVALID_HEVC_SPS_CHROMA_FORMAT_IDC	Invalid Chroma format IDC.
0x3200001c	STATUS_MKV_INVALID_HEVC_SPS_RESERVED	Invalid HEVC reserved SPS.
0x3200001d	STATUS_MKV_MIN_ANNEX_B_CPD_SIZE	Minimum AnnexBb codec private beta value size. For H264, this value must be equal to or greater that 11. For H265, this value must be equal to or greater than 15.
0x3200001e	STATUS_MKV_ANNEXB_CPD_MISSING_NALUS	Missing codec private data in Annex-B NALUs.
0x3200001f	STATUS_MKV_INVALID_ANNEXB_CPD_NALUS	Invalid codec private beta in Annex-B NALUs.

Code	Message	Description / Recommended action
0x32000020	STATUS_MKV_INVALID_TAG_NAME_LENGTH	Invalid tag name length. Valid value is greater than zero and less than 128.
0x32000021	STATUS_MKV_INVALID_TAG_VALUE_LENGTH	Invalid tag value length. Valid value is greater than zero and less than 256.
0x32000022	STATUS_MKV_INVALID_GENERATOR_STATE_TAGS	Invalid generator state tags.
0x32000023	STATUS_MKV_INVALID_AAC_CPD_SAMPLING_FREQUENCY_INDEX	Invalid AAC codec private data sampling frequency index.
0x32000024	STATUS_MKV_INVALID_AAC_CPD_CHANNEL_CONFIG	Invalid AAC codec private data channel configuration.
0x32000025	STATUS_MKV_INVALID_AAC_CPD	Invalid AAC codec private data.
0x32000026	STATUS_MKV_TRACK_INFO_NOT_FOUND	Track information not found.
0x32000027	STATUS_MKV_INVALID_SEGMENT_UUID	Invalid segment UUID.
0x32000028	STATUS_MKV_INVALID_TRACK_UID	Invalid track UID.
0x32000029	STATUS_MKV_INVALID_CLIENT_ID_LENGTH	

Code	Message	Description / Recommended action
0x3200002a	STATUS_MKV_INVALID _AMS_ACM_CPD	
0x3200002b	STATUS_MKV_MISSING _SPS_FROM_H264_CPD	
0x3200002c	STATUS_MKV_MISSING _PPS_FROM_H264_CPD	
0x3200002d	STATUS_MKV_INVALID _PARENT_TYPE	

Error and status codes returned by the Trace library

The following table contains error and status information that are returned by methods in the `Trace` library.

Code	Message
0x10100001	STATUS_MIN_PROFILER_BUFFER

Error and status codes returned by the Utils library

The following table contains error and status information that are returned by methods in the `Utils` library.

Code	Message
0x40000001	STATUS_INVALID_BASE64_ENCODE
0x40000002	STATUS_INVALID_BASE
0x40000003	STATUS_INVALID_DIGIT
0x40000004	STATUS_INT_OVERFLOW

Code	Message
0x40000005	STATUS_EMPTY_STRING
0x40000006	STATUS_DIRECTORY_OPEN_FAILED
0x40000007	STATUS_PATH_TOO_LONG
0x40000008	STATUS_UNKNOWN_DIR_ENTRY_TYPE
0x40000009	STATUS_REMOVE_DIRECTORY_FAILED
0x4000000a	STATUS_REMOVE_FILE_FAILED
0x4000000b	STATUS_REMOVE_LINK_FAILED
0x4000000c	STATUS_DIRECTORY_ACCESS_DENIED
0x4000000d	STATUS_DIRECTORY_MISSING_PATH
0x4000000e	STATUS_DIRECTORY_ENTRY_STAT_ERROR
0x4000000f	STATUS_STRFTIME_FALIED
0x40000010	STATUS_MAX_TIMESTAMP_FORMAT_STR_LEN_EXCEEDED
0x40000011	STATUS_UTIL_MAX_TAG_COUNT
0x40000012	STATUS_UTIL_INVALID_TAG_VERSION
0x40000013	STATUS_UTIL_TAGS_COUNT_NON_ZERO_TAGS_NULL
0x40000014	STATUS_UTIL_INVALID_TAG_NAME_LEN
0x40000015	STATUS_UTIL_INVALID_TAG_VALUE_LEN

Code	Message
0x4000002a	STATUS_EXPONENTIAL_BACKOFF_INVALID_STATE
0x4000002b	STATUS_EXPONENTIAL_BACKOFF_RETRIES_EXHAUSTED
0x4000002c	STATUS_THREADPOOL_MAX_COUNT
0x4000002d	STATUS_THREADPOOL_INTERNAL_ERROR
0x40100001	STATUS_HASH_KEY_NOT_PRESENT
0x40100002	STATUS_HASH_KEY_ALREADY_PRESENT
0x40100003	STATUS_HASH_ENTRY_ITERATION_ABORT
0x41000001	STATUS_BIT_READER_OUT_OF_RANGE
0x41000002	STATUS_BIT_READER_INVALID_SIZE
0x41100001	STATUS_TIMER_QUEUE_STOP_SCHEDULING
0x41100002	STATUS_INVALID_TIMER_COUNT_VALUE
0x41100003	STATUS_INVALID_TIMER_PERIOD_VALUE
0x41100004	STATUS_MAX_TIMER_COUNT_REACHED
0x41100005	STATUS_TIMER_QUEUE_SHUTDOWN
0x41200001	STATUS_SEMAPHORE_OPERATION_AFTER_SHUTDOWN
0x41200002	STATUS_SEMAPHORE_ACQUIRE_WHEN_LOCKED

Code	Message
0x41300001	STATUS_FILE_LOGGER_INDEX_FILE_INVALID_SIZE

Error and status codes returned by the View library

The following table contains error and status information that are returned by methods in the `View` library.

Code	Message	Description
0x30000001	STATUS_MIN_CONTENT_VIEW_ITEMS	An invalid content view item count was specified. For more information, see Producer SDK limits.
0x30000002	STATUS_INVALID_CONTENT_VIEW_DURATION	An invalid content view duration was specified. For more information, see Producer SDK limits.
0x30000003	STATUS_CONTENT_VIEW_NO_MORE_ITEMS	An attempt was made to get past the head position.
0x30000004	STATUS_CONTENT_VIEW_INVALID_INDEX	An invalid index is specified.
0x30000005	STATUS_CONTENT_VIEW_INVALID_TIMESTAMP	There was an invalid timestamp or a timestamp overlap. The frame decoding timestamp should be greater than or equal to the previous frame timestamp, plus the previous frame duration: ``DTS(n) >= DTS(n-1) + Duration(n-1)``.

Code	Message	Description
		This error often indicates an "unstable" encoder. The encoder produces a burst of encoded frames, and their timestamps are smaller than the intra-frame durations. Or the stream is configured to use SDK timestamps, and the frames are sent faster than the frame durations . To help with some "jitter" in the encoder, specify a smaller frame duration in the `StreamInfo.StreamCaps` structure. For example, if the stream is 25 FPS, each frame's duration is 40 ms. However, to handle the encoder "jitter", we recommend that you use half of that frame duration (20 ms). Some streams require more precise control over the timing for error detection.
0x30000006	STATUS_INVALID_CONTENT_VIEW_LENGTH	An invalid content view item data length was specified.

Errors and status codes returned by PutFrame callbacks - C producer library

The following section contains error and status information that are returned by callbacks for the `PutFrame` operation within the C producer library.

Code	Message	Description	Recommended action
0x15000001	STATUS_STOP_CALLBACK_CHAIN	The callback chain has stopped.	
0x15000002	STATUS_MAX_CALLBACK_CHAIN	The maximum callback chain was reached.	
0x15000003	STATUS_INVALID_PLATFORM_CALLBACKS_VERSION	Invalid PlatformCallbacks structure version.	Specify the correct current version of the structure.
0x15000004	STATUS_INVALID_PRODUCER_CALLBACKS_VERSION	Invalid ProducerCallbacks structure version.	Specify the correct current version of the structure.
0x15000005	STATUS_INVALID_STREAM_CALLBACKS_VERSION	Invalid StreamCallbacks structure version.	Specify the correct current version of the structure.
0x15000006	STATUS_INVALID_AUTH_CALLBACKS_VERSION	Invalid AuthCallbacks structure version.	Specify the correct current version of the structure.
0x15000007	STATUS_INVALID_API_CALLBACKS_VERSION	Invalid ApiCallbacks structure version.	Specify the correct current version of the structure.
0x15000008	STATUS_INVALID_AWS	Invalid AwsCredentials structure version.	Specify the correct current version of the structure.

Code	Message	Description	Recommended action
	_CREDENTIALS_VERSION		
0x15000009	STATUS_MAX_REQUEST_HEADER_COUNT	The maximum request header count was reached.	
0x1500000a	STATUS_MAX_REQUEST_HEADER_NAME_LEN	The maximum request header name length was reached.	
0x1500000b	STATUS_MAX_REQUEST_HEADER_VALUE_LEN	The maximum request header value length was reached.	
0x1500000c	STATUS_INVALID_API_CALL_RETURN_JSON	Invalid return JSON for an API call.	
0x1500000d	STATUS_CURL_INIT_FAILED	Curl initialization failed.	
0x1500000e	STATUS_CURL_LIBRARY_INIT_FAILED	Curl lib initialization failed.	
0x1500000f	STATUS_INVALID_DESCRIBE_STREAM_RETURN_JSON	Invalid return JSON for DescribeStream.	
0x15000010	STATUS_HMAC_GENERATION_ERROR	HMAC generation error.	

Code	Message	Description	Recommended action
0x15000011	STATUS_IOT_FAILED	IoT authorization failed.	
0x15000012	STATUS_MAX_ROLE_ALIAS_LEN_EXCEEDED	The maximum role alias length was reached.	Specify a shorter alias length.
0x15000013	STATUS_MAX_USER_AGENT_NAME_POSTFIX_LEN_EXCEEDED	The maximum agent name postfix length was reached.	
0x15000014	STATUS_MAX_CUSTOM_USER_AGENT_LEN_EXCEEDED	The maximum customer user agent length was reached.	
0x15000015	STATUS_INVALID_USER_AGENT_LENGTH	Invalid user agent length.	
0x15000016	STATUS_INVALID_ENDPOINT_CACHING_PERIOD	Invalid endpoint caching period.	Specify a caching period that is less than 24 hours.
0x15000017	STATUS_IOT_EXPIRATION_OCCURS_IN_PAST	IoT expiration timestamp occurs in the past.	
0x15000018	STATUS_IOT_EXPIRATION_PARSING_FAILED	The IoT expiration parsing has failed.	

Code	Message	Description	Recommended action
0x15000019	STATUS_DUPLICATE_PRODUCER_CALLBACK_FREE_FUNC		
0x1500001a	STATUS_DUPLICATE_STREAM_CALLBACK_FREE_FUNC		
0x1500001b	STATUS_DUPLICATE_AUTH_CALLBACK_FREE_FUNC		
0x1500001c	STATUS_DUPLICATE_API_CALLBACK_FREE_FUNC		
0x1500001d	STATUS_FILE_LOGGER_INDEX_FILE_TOO_LARGE		
0x1500001e	STATUS_MAX_IOT_THING_NAME_LENGTH		
0x1500001f	STATUS_IOT_CREATE_LWS_CONTEXT_FAILED		

Code	Message	Description	Recommended action
0x15000020	STATUS_INVALID_CA_CERT_PATH		
0x15000022	STATUS_FILE_CREDENTIAL_PROVIDER_OPEN_FILE_FAILED		
0x15000023	STATUS_FILE_CREDENTIAL_PROVIDER_INVALID_FILE_LENGTH		
0x15000024	STATUS_FILE_CREDENTIAL_PROVIDER_INVALID_FILE_FORMAT		
0x15000026	STATUS_STREAM_BEING_SHUTDOWN		
0x15000027	STATUS_CLIENT_BEING_SHUTDOWN		
0x15000028	STATUS_CONTINUOUS_RETRY_RESET_FAILED		

Code	Message	Description	Recommended action
0x16000001	STATUS_CURL_PERFORM_FAILED		
0x16000002	STATUS_IOT_INVALID_RESPONSE_LENGTH		
0x16000003	STATUS_IOT_NULL_AWS_CREDS		
0x16000004	STATUS_IOT_INVALID_URI_LEN		
0x16000005	STATUS_TIMESTAMP_STRING_UNRECOGNIZED_FORMAT		

Network Abstraction Layer (NAL) adaptation flag reference

This section contains information about available flags for the `StreamInfo.NalAdaptationFlags` enumeration.

The elementary stream in an application can be in either **Annex-B** or **AVCC** format:

- The **Annex-B** format delimits NALUs (Network Abstraction Layer units) with two bytes of zeroes, followed by one or three bytes of zeroes, followed by the number *1* (called a **start code**, for example, 00000001).
- The **AVCC** format also wraps NALUs, but each NALU is preceded by a value that indicates the size of the NALU (usually four bytes).

Many encoders produce the Annex-B bitstream format. Some higher-level bitstream processors (such as a playback engine or the Media Source Extensions (MSE) player in the AWS Management Console) use the AVCC format for their frames.

The codec private data (CPD), which is SPS/PPS (Sequence Parameter Set/Picture Parameter Set) for the H.264 codec, can also be in Annex-B or AVCC format. However, for the CPD, the formats are different from those described previously.

The flags tell the SDK to adapt the NALUs to AVCC or Annex-B for frame data and CPD as follows:

Flag	Adaptation
NAL_ADAPTATION_FLAG_NONE	No adaptation.
NAL_ADAPTATION_ANNEXB_NALS	Adapt Annex-B NALUs to AVCC NALUs.
NAL_ADAPTATION_AVCC_NALS	Adapt AVCC NALUs to Annex-B NALUs.
NAL_ADAPTATION_ANNEXB_CPD_NALS	Adapt Annex-B NALUs for the codec private data to AVCC format NALUs.
NAL_ADAPTATION_ANNEXB_CPD_AND_FRAME_NALS	Adapt Annex-B NALUs for the codec and frame private data to AVCC format NALUs.

For more information about NALU types, see **Section 1.3: Network Abstraction Layer Unit Types** in RFC 3984.

Producer SDK structures

This section includes information about structures that you can use to provide data to the Kinesis Video Streams Producer object.

Topics

- DeviceInfo/DefaultDeviceInfoProvider

- StorageInfo

DeviceInfo/DefaultDeviceInfoProvider

The **DeviceInfo** and **DefaultDeviceInfoProvider** objects control the behavior of the Kinesis Video Streams Producer object.

Member fields

- **version** – An integer value used to make sure that the correct version of the structure is used with the current version of the code base. The current version is specified using the `DEVICE_INFO_CURRENT_VERSION` macro.
- **name** – The human-readable name for the device.
- **tagCount/tags** – Not currently used.
- **streamCount** – The maximum number of streams that the device can handle. This pre-allocates the storage for pointers to the stream objects initially, but the actual stream objects are created later. The default is 16 streams, but you can change this number in the `DefaultDeviceInfoProvider.cpp` file.
- **storageInfo**: An object that describes the main storage configuration. For more information, see StorageInfo.

StorageInfo

Specifies the configuration of the main storage for Kinesis Video Streams.

The default implementation is based on a low-fragmentation fast heap implementation, which is optimized for streaming. It uses the `MEMALLOC` allocator, which can be overwritten on a given platform. Some platforms have virtual memory allocation without backing the allocation with physical pages. As the memory is used, the virtual pages are backed by the physical pages. This results in low-memory pressure on the overall system when storage is underused.

Calculate the default storage size based on the following formula. The `DefragmentationFactor` should be set to 1.2 (20 percent).

```
Size = NumberOfStreams * AverageFrameSize * FramesPerSecond * BufferDurationInSeconds *
 DefragmentationFactor
```

In the following example, a device has audio and video streams. The audio stream has 512 samples per second, with an average sample of 100 bytes. The video stream has 25 frames per second, with an average of 10,000 bytes. Each stream has 3 minutes of buffer duration.

```
Size = (512 * 100 * (3 * 60) + 25 * 10000 * (3 * 60)) * 1.2 = (9216000 + 45000000) *
 1.2 = 65059200 = ~ 66MB.
```

If the device has more available memory, we recommend that you add more memory to storage to avoid severe fragmentation.

Verify that the storage size is adequate to accommodate the full buffers for all streams at high encoding complexity (when the frame size is larger due to high motion) or when the bandwidth is low. If the producer reaches memory pressure, it emits storage overflow pressure callbacks (`StorageOverflowPressureFunc`). However, when no memory is available in the content store, it drops the frame that's being pushed into Kinesis Video Streams with an error (`STATUS_STORE_OUT_OF_MEMORY = 0x5200002e`). For more information, see [Error and status codes returned by the client library](). This can also happen if the application acknowledgements (ACKs) are not available, or the persisted ACKs are delayed. In this case, the buffers fill to the "buffer duration" capacity before the previous frames start dropping out.

Member fields

- **version** – An integer value used to make sure that the correct version of the structure is used with the current version of the code base.
- **storageType** – A `DEVICE_STORAGE_TYPE` enumeration that specifies the underlying backing and implementation of the storage. Currently the only supported value is `DEVICE_STORAGE_TYPE_IN_MEM`. A future implementation will support `DEVICE_STORAGE_TYPE_HYBRID_FILE`, indicating that storage falls back to the file-backed content store.
- **storageSize** – The storage size in bytes to preallocate. The minimum allocation is 10 MB, and the maximum allocation is 10 GB. (This will change with the future implementation of the file-backed content store.)
- **spillRatio** – An integer value that represents the percentage of the storage to be allocated from the direct memory storage type (RAM), as opposed to the secondary overflow storage (file storage). Not currently used.
- **rootDirectory**: The path to the directory where the file-backed content store is located. Not currently used.

Kinesis video stream structures

You can use the following structures to provide data to an instance of a Kinesis video stream.

Topics

- StreamDefinition/StreamInfo
- ClientMetrics
- StreamMetrics

StreamDefinition/StreamInfo

The `StreamDefinition` object in the C++ layer wraps the `StreamInfo` object in the platform-independent code, and provides some default values in the constructor.

Member fields

Field	Data type	Description	Default value
stream_name	`string`	An optional stream name. For more information about the length of the stream name, see Producer SDK limits . Each stream should have a unique name.	If no name is specified, a name is generated randomly.
retention_period	`duration< uint64_t, ratio<3600>>`	The retention period for the stream, in seconds. Specifying **`0`** indicates no retention .	3600 (One hour)
tags	`const map<string, string>*`	A map of key-value pairs that contain user information. If the stream already	No tags

Field	Data type	Description	Default value
		has a set of tags, the new tags are appended to the existing set of tags.	
kms_key_id	`string`	The AWS KMS key ID to be used for encrypting the stream. For more information, see Data protection in Kinesis Video Streams.	The default KMS key (`aws/kinesis-video`.)
streaming_type	`STREAMING_TYPE` enumeration	The only supported value is `STREAMING_TYPE_REALTIME`.	
content_type	`string`	The content format of the stream. The Kinesis Video Streams console can play back content in the `video/h264` format.	`video/h264`

Field	Data type	Description	Default value
max_latency	`duration<uint64_t, milli>`	The maximum latency in milliseconds for the stream. The stream latency pressure callback (if specified) is called when the buffer duration exceeds this amount of time. Specifying **`0`** indicates that no stream latency pressure callback will be called.	`milliseconds::zero()`

Field	Data type	Description	Default value
fragment_duration	`duration<uint64_t>`	The fragment duration that you want, in seconds. This value is used in combination with the `key_frame_fragmentation` value. If this value is `false`, Kinesis Video Streams generates fragments on a key frame after this duration elapses. For example, an Advanced Audio Coding (AAC) audio stream has each frame as a key frame. Specifying `key_frame_fragmentation = false` causes fragmentation to happen on a key frame after this duration expires, resulting in 2-second fragments.	2

Field	Data type	Description	Default value
timecode_scale	`duration<uint64_t, milli>`	The MKV timecode scale in milliseconds, which specifies the granularity of the timecodes for the frames within the MKV cluster. The MKV frame timecode is always relative to the start of the cluster. MKV uses a signed 16-bit value (0-32767) to represent the timecode within the cluster (fragment). Verify that the frame timecode can be represented with the given timecode scale. The default timecode scale value of 1 ms ensures that the largest frame that can be represented is 32767 ms ~= 32 seconds. This is over the maximum fragment duration that is specified in [Kinesis Video Streams service quotas](), which is 10 seconds.	1

Field	Data type	Description	Default value
key_frame_fragmentation	`bool`	Whether to produce fragments on a key frame. If `true`, the SDK produces a start of the fragment every time there is a key frame. If `false`, Kinesis Video Streams waits for at least `fragment_duration` and produces a new fragment on the key frame following it.	`true`
frame_timecodes	`bool`	Whether to use frame timecodes or generate timestamps using the current time callback. Many encoders don't produce timestamps with the frames. So specifying `false` for this parameter ensures that the frames are timestamped as they are put into Kinesis Video Streams.	`true`

Field	Data type	Description	Default value
absolute_fragment_times	`bool`	Kinesis Video Streams uses MKV as its underlying packaging mechanism. The MKV specification is strict about frame timecodes being relative to the beginning of the cluster (fragment). However, the cluster timecodes can be either absolute or relative to the starting time for the stream. If the timestamps are relative, the `PutMedia` service API call uses the optional stream start timestamp and adjust the cluster timestamp s. The service always stores the fragments with their absolute timestamps.	`true`
fragment_acks	`bool`	Whether to receive application level fragment ACKs (acknowledgements).	`true`, meaning that the SDK will receive the ACKs and act accordingly.

Field	Data type	Description	Default value
restart_on_error	`bool`	Whether to restart on specific errors.	`true`, meaning that the SDK tries to restart the streaming if any errors occur.
recalculate_metrics	`bool`	Whether to recalcula te the metrics. Each call to retrieve the metrics can recalcula te those to get the latest "running" value, which might create a small CPU impact. You might need to set this to `false` on extremely low-power/footprin t devices to spare the CPU cycles. Otherwise, we don't recommend using `false` for this value.	`true`

Field	Data type	Description	Default value
nal_adaptation_flags	`uint32_t`	Specifies the Network Abstraction Layer unit (NALU) adaptation flags. If the bitstream is H.264 encoded, it can then be processed as raw or packaged in NALUs. Those are either in the Annex-B or AVCC format. Most of the elementary stream producers and consumers (read encoders and decoders) use the Annex-B format because it has advantages, such as error recovery. Higher-level systems use the AVCC format, which is the default format for MPEG, HLS, DASH, and so on. The console playback uses the browser's MSE (media source extensions) to decode and play back the stream that uses the AVCC format. For H.264 (and for M-JPEG and H.265),	The default is to adapt Annex-B format to AVCC format for both the frame data and for the codec private data.

Field	Data type	Description	Default value
		the SDK provides adaptation capabilities. Many elementary streams are in the following format. In this example, Ab is the Annex-B start code (001 or 0001). `Ab(Sps)Ab(Pps)Ab(I-frame)Ab(P/B-frame) Ab(P/B-frame)…. Ab(Sps)Ab(Pps)Ab(I-frame)Ab(P/B-frame) Ab(P/B-frame)` In the case of H.264, the codec private data (CPD) is in the SPS (sequence parameter set) and PPS (picture parameter set) parameters, and can be adapted to the AVCC format. Unless the media pipeline gives the CPD separately, the application can extract the CPD from	

Field	Data type	Description	Default value
		the frame. It can do this by looking for the first IDR frame (which should contain the SPS and PPS), extract the two NALUs (which are `Ab(Sps)Ab(Pps)`), and set it in the CPD in `StreamDefinition` . For more information, see [NAL Adaptation Flags]().	
frame_rate	`uint32_t`	The expected frame rate. This value is used to better calculate buffering needs.	25
avg_bandwidth_bps	`uint32_t`	The expected average bandwidth for the stream. This value is used to better calculate buffering needs.	4 * 1024 * 1024

Field	Data type	Description	Default value
buffer_duration	`duration<uint64_t>`	The stream buffer duration, in seconds. The SDK keeps the frames in the content store for up to the `buffer_duration` , after which the previous frames are dropped as the window moves forward. If the frame that's being dropped hasn't been sent to the backend, the dropped frame callback is called. If the current buffer duration is greater than `max_latency` , then the stream latency pressure callback is called. The buffer is trimmed to the next fragment start when the fragment persisted ACK is received. This indicates that the content has been durably persisted in the cloud, so storing the content on the local device is no longer needed.	120

Field	Data type	Description	Default value
replay_duration	`duration<uint64_t>`	The duration, in seconds, to roll the current reader backward to replay during an error if restarting is enabled. The rollback stops at the buffer start (in case it has just started streaming or the persisted ACK has come along). The rollback tries to land on a key frame that indicates a fragment start. If the error that ' causing the restart isn't indicative of a dead host (the host is still alive and contains the frame data in its internal buffers), the rollback stops at the last received ACK frame. It then rolls forward to the next key frame, because the entire fragment is already stored in the host memory.	40

Field	Data type	Description	Default value
connection_staleness	`duration<uint64_t>`	The time, in seconds, after which the stream staleness callback is called if the SDK doesn't receive the buffering ACK. It indicates that the frames are being sent from the device, but the backend isn't acknowledging them. This condition indicates a severed connection at the intermediate hop or at the load balancer.	30
codec_id	`string`	The codec ID for the MKV track.	`V_MPEG4/ISO/AVC`
track_name	`string`	The MKV track name.	`kinesis_video`

Field	Data type	Description	Default value
codecPrivateData	`unsigned char*`	The codec private data (CPD) buffer. If the media pipeline has the informati on about the CPD before the stream starts, it can be set in `StreamDefinition.codecPrivateData` . The bits are copied, and the buffer can be reused or freed after the call to create the stream. However, if the data isn't available when the stream is created, it can be set in one of the overloads of the `KinesisVideoStream.start(cpd)` function.	`null`
codecPrivateDataSize	`uint32_t`	The codec private data buffer size.	0

ClientMetrics

The **ClientMetrics** object is filled by calling `getKinesisVideoMetrics`.

Member fields

Field	Data type	Description
version	UINT32	The version of the structure , defined in the CLIENT_METRICS_CURRENT_VERSION macro.
contentStoreSize	UINT64	The overall content store size in bytes. This is the value specified in DeviceInfo.StorageInfo.storageSize .
contentStoreAvailableSize	UINT64	Current available storage size in bytes.
contentStoreAllocatedSize	UINT64	Current allocated size. The allocated plus the available sizes should be slightly smaller than the overall storage size, due to the internal bookkeeping and the implementation of the content store.
totalContentViewsSize	UINT64	The size of the memory allocated for all content views for all streams. This isn't counted against the storage size. This memory is allocated using the MEMALLOC macro, which can be overwritten to provide a custom allocator.
totalFrameRate	UINT64	The total observed frame rate across all the streams.

Field	Data type	Description
totalTransferRate	UINT64	The total observed stream rate in bytes per second across all the streams.

StreamMetrics

The **StreamMetrics** object is filled by calling `getKinesisVideoMetrics`.

Member fields

Field	Data type	Description
version	UINT32	The version of the structure, defined in the `STREAM_METRICS_CURRENT_VERSION` macro.
currentViewDuration	UINT64	The duration of the accumulated frames. In the fast networking case, this duration is either zero or the frame duration (as the frame is being transmitted). If the duration becomes longer than the `max_latency` specified in the `StreamDefinition`, the stream latency callback is called if it's specified. The duration is specified in 100 ns units, which is the default time unit for the PIC layer.
overallViewDuration	UINT64	The overall view duration. If the stream is configured with no ACKs or persistence, this

Field	Data type	Description
		value grows as the frames are put into the Kinesis video stream and becomes equal to the `buffer_duration` in the `StreamDefinition`. When ACKs are enabled and the persisted ACK is received, the buffer is trimmed to the next key frame. This is because the ACK timestamp indicates the beginning of the entire fragment. The duration is specified in 100-ns units, which is the default time unit for the PIC layer.
currentViewSize	`UINT64`	The size in bytes of the current buffer.
overallViewSize	`UINT64`	The overall view size in bytes.
currentFrameRate	`UINT64`	The observed frame rate for the current stream.
currentTransferRate	`UINT64`	The observed transfer rate in bytes per second for the current stream.

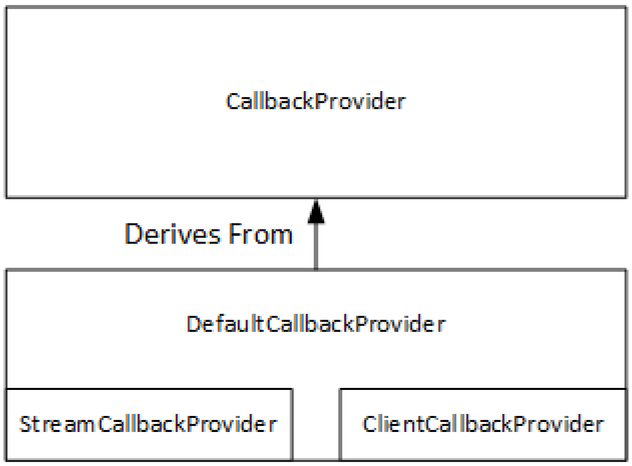
Producer SDK callbacks

The classes and methods in the Amazon Kinesis Video Streams Producer SDK don't maintain their own processes. Instead, they use the incoming function calls and events to schedule callbacks to communicate with the application.

There are two callback patterns that the application can use to interact with the SDK:

- `CallbackProvider` – This object exposes every callback from the platform-independent code (PIC) component to the application. This pattern allows full functionality, but it also means that the implementation must handle all of the public API methods and signatures in the C++ layer.
- StreamCallbackProvider and ClientCallbackProvider – These objects expose the stream-specific and client-specific callbacks, and the C++ layer of the SDK exposes the rest of the callbacks. This is the preferred callback pattern for interacting with the Producer SDK.

The following diagram illustrates the object model of the callback objects:



In the preceding diagram, `DefaultCallbackProvider` derives from `CallbackProvider` (which exposes all of the callbacks in the PIC) and contains `StreamCallbackProvider` and `ClientCallbackProvider`.

This topic contains the following sections:

- ClientCallbackProvider
- StreamCallbackProvider
- ClientCallbacks structure
- Callback implementations to retry streaming

ClientCallbackProvider

The `ClientCallbackProvider` object exposes client-level callback functions. The details of the functions are described in the ClientCallbacks section.

Callback methods:

- `getClientReadyCallback` – Reports a ready state for the client.

- getStorageOverflowPressureCallback – Reports storage overflow or pressure. This callback is called when the storage utilization drops below the STORAGE_PRESSURE_NOTIFICATION_THRESHOLD value, which is 5 percent of the overall storage size. For more information, see StorageInfo.

StreamCallbackProvider

The StreamCallbackProvider object exposes stream-level callback functions.

Callback methods:

- getDroppedFragmentReportCallback: Reports a dropped fragment.
- getDroppedFrameReportCallback – Reports a dropped frame.
- getFragmentAckReceivedCallback – Reports that a fragment ACK is received for the stream.
- getStreamClosedCallback – Reports a stream closed condition.
- getStreamConnectionStaleCallback – Reports a stale connection condition. In this condition, the producer is sending data to the service but isn't receiving acknowledgements.
- getStreamDataAvailableCallback – Reports that data is available in the stream.
- getStreamErrorReportCallback – Reports a stream error condition.
- getStreamLatencyPressureCallback – Reports a stream latency condition, which is when the accumulated buffer size is larger than the max_latency value. For more information, see StreamDefinition/StreamInfo.
- getStreamReadyCallback: –Reports a stream ready condition.
- getStreamUnderflowReportCallback – Reports a stream underflow condition. This function isn't currently used and is reserved for future use.

For the source code for StreamCallbackProvider, see StreamCallbackProvider.h.

ClientCallbacks structure

The ClientCallbacks structure contains the callback function entry points that the PIC calls when specific events occur. The structure also contains version information in the CALLBACKS_CURRENT_VERSION field, and a customData field for user-defined data that is returned with the individual callback functions.

The client application can use a `this` pointer for the `custom_data` field to map member functions to the static `ClientCallback` functions at runtime, as shown in the following code example:

```
STATUS TestStreamCallbackProvider::streamClosedHandler(UINT64 custom_data,
 STREAM_HANDLE stream_handle, UINT64 stream_upload_handle) {
    LOG_INFO("Reporting stream stopped.");

TestStreamCallbackProvider* streamCallbackProvider =
 reinterpret_cast<TestStreamCallbackProvider*> (custom_data);
streamCallbackProvider->streamClosedHandler(...);
```

Events

Function	Description	Type
`CreateDeviceFunc`	Not currently implemented on the backend. This call fails when called from Java or C ++. Other clients perform platform-specific initializ ation.	Backend API
`CreateStreamFunc`	Called when the stream is created.	Backend API
`DescribeStreamFunc`	Called when `DescribeS tream` is called.	Backend API
`GetStreamingEndpoi ntFunc`	Called when `GetStream ingEndpoint` is called.	Backend API
`GetStreamingTokenF unc`	Called when `GetStream ingToken` is called.	Backend API
`PutStreamFunc`	Called when `PutStream` is called.	Backend API
`TagResourceFunc`	Called when `TagResource` is called.	Backend API

Function	Description	Type
`CreateMutexFunc`	Creates a synchronization mutex.	Synchronization
`FreeMutexFunc`	Frees the mutex.	Synchronization
`LockMutexFunc`	Locks the synchronization mutex.	Synchronization
`TryLockMutexFunc`	Tries to lock the mutex. Not currently implemented.	Synchronization
`UnlockMutexFunc`	Unlocks the mutex.	Synchronization
`ClientReadyFunc`	Called when the client enters a ready state.	Notification
`DroppedFrameReportFunc`	Reports when a frame is dropped.	Notification
`DroppedFragmentReportFunc`	Reports when a fragment is dropped. This function isn't currently used and is reserved for future use.	Notification
`FragmentAckReceivedFunc`	Called when a fragment ACK (buffering, received, persisted, and error) is received.	Notification

Function	Description	Type
`StorageOverflowPressureFunc`	Called when the storage utilization drops below the `STORAGE_PRESSURE_NOTIFICATION_THRESHOLD` value, which is defined as 5 percent of the overall storage size.	Notification
`StreamClosedFunc`	Called when the last bits of the remaining frames are streamed.	Notification
`StreamConnectionStaleFunc`	Called when the stream enters a stale connection state. In this condition, the producer is sending data to the service but is not receiving acknowledgements.	Notification
`StreamDataAvailableFunc`	Called when stream data is available.	Notification
`StreamErrorReportFunc`	Called when a stream error occurs. The PIC automatically closes the stream under this condition.	Notification
`StreamLatencyPressureFunc`	Called when the stream enters a latency condition, which is when the accumulated buffer size is larger than the `max_latency` value. For more information, see [StreamDefinition/StreamInfo]().	Notification

Function	Description	Type
StreamReadyFunc	Called when the stream enters the ready state.	Notification
StreamUnderflowReportFunc	This function isn't currently used and is reserved for future use.	Notification
DeviceCertToTokenFunc	Returns the connection certificate as a token.	Platform integration
GetCurrentTimeFunc	Returns the current time.	Platform integration
GetDeviceCertificateFunc	Returns the device certificate. This function isn't currently used and is reserved for future use.	Platform integration
GetDeviceFingerprintFunc	Returns the device fingerprint. This function isn't currently used and is reserved for future use.	Platform integration
GetRandomNumberFunc	Returns a random number between 0 and RAND_MAX.	Platform integration
GetSecurityTokenFunc	Returns the security token that's passed to the functions that communicate with the backend API. The implementation can specify the serialized AccessKeyId , SecretKeyId , and the session token.	Platform integration

Function	Description	Type
`LogPrintFunc`	Logs a line of text with the tag and the log level. For more information, see `PlatformUtils.h` .	Platform integration

For the platform integration functions in the preceding table, the last parameter is a `ServiceCallContext` structure, which has the following fields:

- `version`: The version of the struct.
- `callAfter`: An absolute time after which to call the function.
- `timeout`: The timeout of the operation in 100 nanosecond units.
- `customData`: A user-defined value to be passed back to the client.
- `pAuthInfo`: The credentials for the call. For more information, see the following (`__AuthInfo`) structure.

The authorization information is provided using the `__AuthInfo` structure, which can be either serialized credentials or a provider-specific authentication token. This structure has the following fields:

- `version`: The version of the `__AuthInfo` structure.
- `type`: An `AUTH_INFO_TYPE` value defining the type of the credential (certificate or security token).
- `data`: A byte array containing the authentication information.
- `size`: The size of the `data` parameter.
- `expiration`: The expiration of the credentials in 100 nanosecond units.

Callback implementations to retry streaming

The Kinesis Video Producer SDK provides the status of streaming through callback functions. We recommend that you implement the following callback mechanisms to recover from any momentary network issues encountered during streaming.

- **Stream latency pressure callback** - this callback mechanism gets initiated when the SDK encounters a stream latency condition. This happens when the accumulated buffer size is larger than the MAX_LATENCY value. When the stream is created, the streaming application sets MAX_LATENCY to the default value of 60 seconds. The typical implementation for this callback is to reset the connection. You can use the sample implementation at https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp/blob/master/kinesis-video-c-producer/src/source/StreamLatencyStateMachine.c as needed. Note that there's no option to store the frames undelivered due to network outage into a secondary storage for back-fill.
- **Stream staleness callback** - this callback gets initiated when the producer can send data to the Amazon Kinesis Data Streams service (uplink) but it’s not able to get the acknowledgements (buffered ACK) back in time (default is 60 seconds). Depending on the network settings, either the stream latency pressure callback or the stream staleness callback, or both can get initiated. Similar to the stream latency pressure callback retry implementation, the typical implementation is to reset the connection and start a new connection for streaming. You can use the sample implementation at https://github.com/awslabs/amazon-kinesis-video-streams-producer-c/blob/master/src/source/ConnectionStaleStateMachine.c as needed.
- **Stream error callback** - this callback gets initiated when the SDK encounters a timeout on the network connection or other errors during the call to the KVS API service calls.
- **Dropped frame callback** - this callback gets initiated when the storage size is full either because of slow network speed or a stream error. If the network speed results in dropped frames, you can either increase the storage size, reduce the video frame size, or frame rate to match the network speed.

Kinesis video stream Parser Library

The Kinesis video stream Parser Library is a set of tools that you can use in Java applications to consume the MKV data in a Kinesis video stream.

The library includes the following tools:

- StreamingMkvReader: This class reads specified MKV elements from a video stream.
- FragmentMetadataVisitor: This class retrieves metadata for fragments (media elements) and tracks (individual data streams containing media information, such as audio or subtitles).
- OutputSegmentMerger: This class merges consecutive fragments or chunks in a video stream.
- KinesisVideoExample: This is a sample application that shows how to use the Kinesis video stream Parser Library.

The library also includes tests that show how the tools are used.

Procedure: Using the Kinesis video stream Parser Library

This procedure includes the following steps:

- the section called “Step 1: Download and configure the code”.
- the section called “Step 2: Write and examine the code”.
- the section called “Step 3: Run and verify the code”.

Prerequisites

You must have the following to examine and use the Kinesis video stream Parser Library:

- An Amazon Web Services (AWS) account. If you don't already have an AWS account, see the section called “Sign up for an AWS account”.
- A Java integrated development environment (IDE), such as Eclipse Java Neon or JetBrains IntelliJ Idea.
- Java 11, such as Amazon Corretto 11.

Step 1: Download and configure the code

In this section, you download the Java library and test code, and import the project into your Java IDE.

For prerequisites and other details about this procedure, see *Stream Parser Library*.

1. Create a directory and clone the library source code from the GitHub repository (https://github.com/aws/amazon-kinesis-video-streams-parser-library).

```
git clone https://github.com/aws/amazon-kinesis-video-streams-parser-library
```

2. Open the Java IDE that you're using (for example, Eclipse or IntelliJ IDEA) and import the Apache Maven project that you downloaded:

   - **In Eclipse:** Choose **File**, **Import**, **Maven**, **Existing Maven Projects**, and navigate to the `kinesis-video-streams-parser-lib` folder.
   - **In IntelliJ Idea:** Choose **Import**. Navigate to the **pom.xml** file in the root of the downloaded package.

   For more information, see the related IDE documentation.

Next step

the section called “Step 2: Write and examine the code”.

Step 2: Write and examine the code

In this section, you examine the Java library and test code, and learn how to use the tools from the library in your own code.

The Kinesis video stream Parser Library contains the following tools:

- StreamingMkvReader
- FragmentMetadataVisitor
- OutputSegmentMerger
- KinesisVideoExample

StreamingMkvReader

This class reads specified MKV elements from a stream in a non-blocking way.

The following code example (from `FragmentMetadataVisitorTest`) shows how to create and use a `Streaming MkvReader` to retrieve `MkvElement` objects from an input stream called `inputStream`:

```
StreamingMkvReader mkvStreamReader =
                StreamingMkvReader.createDefault(new
 InputStreamParserByteSource(inputStream));
        while (mkvStreamReader.mightHaveNext()) {
            Optional<MkvElement> mkvElement = mkvStreamReader.nextIfAvailable();
            if (mkvElement.isPresent()) {
                mkvElement.get().accept(fragmentVisitor);
                ...
                }
            }
        }
```

FragmentMetadataVisitor

This class retrieves metadata for fragments (media elements) and tracks individual data streams containing media information, such as codec private data, pixel width, or pixel height.

The following code example (from the `FragmentMetadataVisitorTest` file) shows how to use `FragmentMetadataVisitor` to retrieve data from a `MkvElement` object:

```
FragmentMetadataVisitor fragmentVisitor = FragmentMetadataVisitor.create();
        StreamingMkvReader mkvStreamReader =
                StreamingMkvReader.createDefault(new InputStreamParserByteSource(in));
        int segmentCount = 0;
        while(mkvStreamReader.mightHaveNext()) {
            Optional<MkvElement> mkvElement = mkvStreamReader.nextIfAvailable();
            if (mkvElement.isPresent()) {
                mkvElement.get().accept(fragmentVisitor);
                if
 (MkvTypeInfos.SIMPLEBLOCK.equals(mkvElement.get().getElementMetaData().getTypeInfo()))
 {
                    MkvDataElement dataElement = (MkvDataElement) mkvElement.get();
                    Frame frame =
 ((MkvValue<Frame>)dataElement.getValueCopy()).getVal();
```

```
                    MkvTrackMetadata trackMetadata =
fragmentVisitor.getMkvTrackMetadata(frame.getTrackNumber());
                    assertTrackAndFragmentInfo(fragmentVisitor, frame, trackMetadata);
                }
                if
(MkvTypeInfos.SEGMENT.equals(mkvElement.get().getElementMetaData().getTypeInfo())) {
                    if (mkvElement.get() instanceof MkvEndMasterElement) {
                        if (segmentCount < continuationTokens.size()) {
                            Optional<String> continuationToken =
fragmentVisitor.getContinuationToken();
                            Assert.assertTrue(continuationToken.isPresent());
                            Assert.assertEquals(continuationTokens.get(segmentCount),
continuationToken.get());
                        }
                        segmentCount++;
                    }
                }
            }

        }
```

The preceding example shows the following coding pattern:

- Create a `FragmentMetadataVisitor` to parse the data, and a [StreamingMkvReader](#) to provide the data.
- For each `MkvElement` in the stream, test if its metadata is of type `SIMPLEBLOCK`.
- If it is, retrieve the `MkvDataElement` from the `MkvElement`.
- Retrieve the `Frame` (media data) from the `MkvDataElement`.
- Retrieve the `MkvTrackMetadata` for the `Frame` from the `FragmentMetadataVisitor`.
- Retrieve and verify the following data from the `Frame` and `MkvTrackMetadata` objects:
  - The track number.
  - The frame's pixel height.
  - The frame's pixel width.
  - The codec ID for the codec used to encode the frame.
  - That this frame arrived in order. Verify that the track number of the previous frame, if present, is less than that of the current frame.

To use `FragmentMetadataVisitor` in your project, pass `MkvElement` objects to the visitor using their `accept` method:

```
mkvElement.get().accept(fragmentVisitor);
```

OutputSegmentMerger

This class merges metadata from different tracks in the stream into a stream with a single segment.

The following code example (from the `FragmentMetadataVisitorTest` file) shows how to use `OutputSegmentMerger` to merge track metadata from a byte array called `inputBytes`:

```
FragmentMetadataVisitor fragmentVisitor = FragmentMetadataVisitor.create();

ByteArrayOutputStream outputStream = new ByteArrayOutputStream();

OutputSegmentMerger outputSegmentMerger =
    OutputSegmentMerger.createDefault(outputStream);

CompositeMkvElementVisitor compositeVisitor =
    new TestCompositeVisitor(fragmentVisitor, outputSegmentMerger);

final InputStream in = TestResourceUtil.getTestInputStream("output_get_media.mkv");

StreamingMkvReader mkvStreamReader =
    StreamingMkvReader.createDefault(new InputStreamParserByteSource(in));

while (mkvStreamReader.mightHaveNext()) {
    Optional<MkvElement> mkvElement = mkvStreamReader.nextIfAvailable();
    if (mkvElement.isPresent()) {
        mkvElement.get().accept(compositeVisitor);
    if
 (MkvTypeInfos.SIMPLEBLOCK.equals(mkvElement.get().getElementMetaData().getTypeInfo()))
 {
        MkvDataElement dataElement = (MkvDataElement) mkvElement.get();
        Frame frame = ((MkvValue<Frame>) dataElement.getValueCopy()).getVal();
        Assert.assertTrue(frame.getFrameData().limit() > 0);
        MkvTrackMetadata trackMetadata =
 fragmentVisitor.getMkvTrackMetadata(frame.getTrackNumber());
        assertTrackAndFragmentInfo(fragmentVisitor, frame, trackMetadata);
    }
```

```
}
```

The preceding example shows the following coding pattern:

- Create a FragmentMetadataVisitorto retrieve the metadata from the stream.
- Create an output stream to receive the merged metadata.
- Create an `OutputSegmentMerger`, passing in the `ByteArrayOutputStream`.
- Create a `CompositeMkvElementVisitor` that contains the two visitors.
- Create an `InputStream` that points to the specified file.
- Merge each element in the input data into the output stream.

KinesisVideoExample

This is a sample application that shows how to use the Kinesis video stream Parser Library.

This class performs the following operations:

- Creates a Kinesis video stream. If a stream with the given name already exists, the stream is deleted and recreated.
- Calls PutMedia to stream video fragments to the Kinesis video stream.
- Calls GetMedia to stream video fragments out of the Kinesis video stream.
- Uses a StreamingMkvReader to parse the returned fragments on the stream, and uses a FragmentMetadataVisitor to log the fragments.

Delete and recreate the stream

The following code example (from the `StreamOps.java` file) deletes a given Kinesis video stream:

```
//Delete the stream
amazonKinesisVideo.deleteStream(new
 DeleteStreamRequest().withStreamARN(streamInfo.get().getStreamARN()));
```

The following code example (from the `StreamOps.java` file) creates a Kinesis video stream with the specified name:

```
amazonKinesisVideo.createStream(new CreateStreamRequest().withStreamName(streamName)
.withDataRetentionInHours(DATA_RETENTION_IN_HOURS)
.withMediaType("video/h264"));
```

Call PutMedia

The following code example (from the `PutMediaWorker.java` file) calls PutMedia on the stream:

```
 putMedia.putMedia(new PutMediaRequest().withStreamName(streamName)
.withFragmentTimecodeType(FragmentTimecodeType.RELATIVE)
.withProducerStartTimestamp(new Date())
.withPayload(inputStream), new PutMediaAckResponseHandler() {
...
});
```

Call GetMedia

The following code example (from the `GetMediaWorker.java` file) calls GetMedia on the stream:

```
GetMediaResult result = videoMedia.getMedia(new
 GetMediaRequest().withStreamName(streamName).withStartSelector(startSelector));
```

Parse the GetMedia result

This section describes how to use StreamingMkvReader, FragmentMetadataVisitor and `CompositeMkvElementVisitor` to parse, save to file, and log the data returned from `GetMedia`.

Read the output of GetMedia with StreamingMkvReader

The following code example (from the `GetMediaWorker.java` file) creates a StreamingMkvReader and uses it to parse the result from the GetMedia operation:

```
StreamingMkvReader mkvStreamReader = StreamingMkvReader.createDefault(new
 InputStreamParserByteSource(result.getPayload()));
log.info("StreamingMkvReader created for stream {} ", streamName);
try {
    mkvStreamReader.apply(this.elementVisitor);
} catch (MkvElementVisitException e) {
    log.error("Exception while accepting visitor {}", e);
```

```
}
```

In the preceding code example, the StreamingMkvReader retrieves `MKVElement` objects from the payload of the `GetMedia` result. In the next section, the elements are passed to a FragmentMetadataVisitor.

Retrieve fragments with FragmentMetadataVisitor

The following code examples (from the `KinesisVideoExample.java` and `StreamingMkvReader.java` files) create a FragmentMetadataVisitor. The `MkvElement` objects iterated by the StreamingMkvReader are then passed to the visitor using the `accept` method.

from `KinesisVideoExample.java`:

```
FragmentMetadataVisitor fragmentMetadataVisitor = FragmentMetadataVisitor.create();
```

from `StreamingMkvReader.java`:

```
if (mkvElementOptional.isPresent()) {
    //Apply the MkvElement to the visitor
    mkvElementOptional.get().accept(elementVisitor);
        }
```

Log the elements and write them to a file

The following code example (from the `KinesisVideoExample.java` file) creates the following objects and returns them as part of the return value of the `GetMediaProcessingArguments` function:

- A `LogVisitor` (an extension of `MkvElementVisitor`) that writes to the system log.
- An `OutputStream` that writes the incoming data to an MKV file.
- A `BufferedOutputStream` that buffers data bound for the `OutputStream`.
- An the section called “OutputSegmentMerger” that merges consecutive elements in the `GetMedia` result with the same track and EBML data.
- A `CompositeMkvElementVisitor` that composes the FragmentMetadataVisitor, the section called “OutputSegmentMerger”, and `LogVisitor` into a single element visitor.

```
//A visitor used to log as the GetMedia stream is processed.
    LogVisitor logVisitor = new LogVisitor(fragmentMetadataVisitor);

    //An OutputSegmentMerger to combine multiple segments that share track and ebml
 metadata into one
    //mkv segment.
    OutputStream fileOutputStream =
 Files.newOutputStream(Paths.get("kinesis_video_example_merged_output2.mkv"),
            StandardOpenOption.WRITE, StandardOpenOption.CREATE);
    BufferedOutputStream outputStream = new BufferedOutputStream(fileOutputStream);
    OutputSegmentMerger outputSegmentMerger =
 OutputSegmentMerger.createDefault(outputStream);

    //A composite visitor to encapsulate the three visitors.
    CompositeMkvElementVisitor mkvElementVisitor =
            new CompositeMkvElementVisitor(fragmentMetadataVisitor,
 outputSegmentMerger, logVisitor);

    return new GetMediaProcessingArguments(outputStream, logVisitor,
 mkvElementVisitor);
```

The media processing arguments are then passed into the `GetMediaWorker`, which is in turn passed to the `ExecutorService`, which carries out the worker on a separate thread:

```
GetMediaWorker getMediaWorker = GetMediaWorker.create(getRegion(),
        getCredentialsProvider(),
        getStreamName(),
        new StartSelector().withStartSelectorType(StartSelectorType.EARLIEST),
        amazonKinesisVideo,
        getMediaProcessingArgumentsLocal.getMkvElementVisitor());
executorService.submit(getMediaWorker);
```

Next step

the section called “Step 3: Run and verify the code”

Step 3: Run and verify the code

The Kinesis video stream Parser Library contains tools that are intended for you to use in your own projects. The project contains unit tests for the tools that you can run to verify your installation.

The following unit tests are included in the library:

- **mkv**
  - `ElementSizeAndOffsetVisitorTest`
  - `MkvValueTest`
  - `StreamingMkvReaderTest`
- **utilities**
  - `FragmentMetadataVisitorTest`
  - `OutputSegmentMergerTest`

Amazon Kinesis Video Streams examples

The following code examples demonstrate how to work with the Kinesis Video Streams API:

Examples: Sending data to Kinesis Video Streams

- Example: Kinesis Video Streams Producer SDK GStreamer Plugin - kvssink: Shows how to build the Kinesis Video Streams Producer SDK to use as a GStreamer destination.
- Run the GStreamer element in a Docker container: Shows how to use a pre-built Docker image for sending Real-Time Streaming Protocol (RTSP) video from an IP camera to Kinesis Video Streams.
- Example: Streaming from an RTSP source: Shows how to build your own Docker image and send RTSP video from an IP camera to Kinesis Video Streams.
- Example: Sending data to Kinesis Video Streams using the PutMedia API: Shows how to use the Using the Java Producer Library to send data to Kinesis Video Streams that's already in a container format (MKV) using the PutMedia API.

Examples: Retrieving data from Kinesis Video Streams

- KinesisVideoExample: Shows how to parse and log video fragments using the Kinesis Video Streams Parser Library.
- Example: Parsing and rendering Kinesis Video Streams fragments: Shows how to parse and render Kinesis video stream fragments using JCodec and JFrame.

Examples: Playing back video data

- Example: Use HLS in HTML and JavaScript: Shows how to retrieve an HLS streaming session for a Kinesis video stream and play it back in a webpage.

Prerequisites

- In the sample code, you provide credentials by specifying a profile that you set in your AWS credentials profile file, or by providing credentials in the Java system properties of your integrated development environment (IDE). If you haven't already done so, first set up your credentials. For more information, see Set up AWS Credentials and Region for Development.

- We recommend that you use a Java IDE to view and run the code, such as one of the following:
  - Eclipse Java Neon
  - JetBrains IntelliJ IDEA

Example: Kinesis Video Streams Producer SDK GStreamer Plugin - kvssink

This topic describes how to build the Amazon Kinesis Video Streams Producer SDK to use as a GStreamer plugin.

Topics

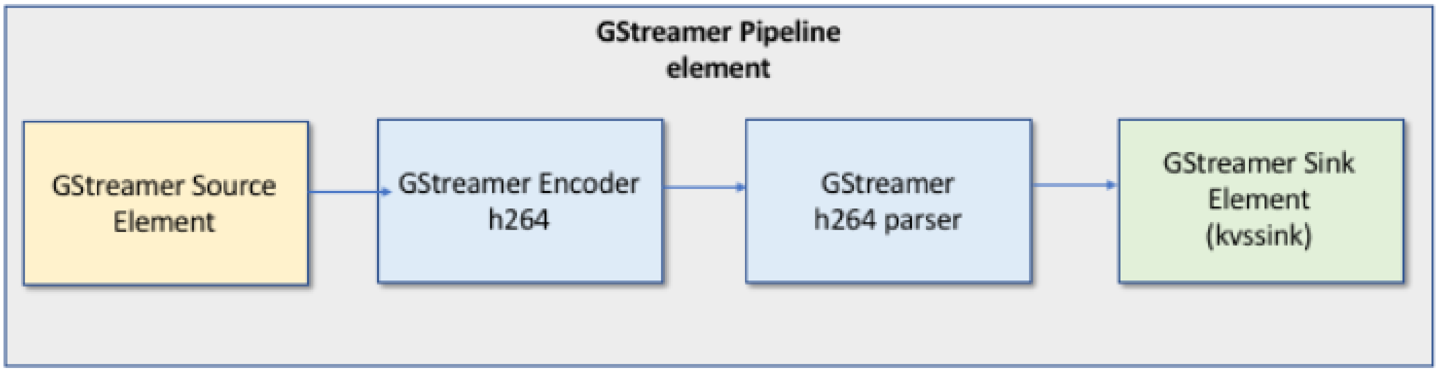
GStreamer is a popular media framework used by multiple cameras and video sources to create custom media pipelines by combining modular plugins. The Kinesis Video Streams GStreamer plugin streamlines the integration of your existing GStreamer media pipeline with Kinesis Video Streams. After integrating GStreamer, you can stream video from a webcam or Real Time Streaming Protocol (RTSP) camera to Kinesis Video Streams for real-time or later playback, storage, and further analysis.

The GStreamer plugin automatically manages the transfer of your video stream to Kinesis Video Streams by encapsulating the functionality provided by the Kinesis Video Streams Producer SDK in a GStreamer sink element, `kvssink`. The GStreamer framework provides a standard managed environment for constructing media flow from a device such as a camera or other video source for further processing, rendering, or storage.

The GStreamer pipeline typically consists of the link between a source (video camera) and the sink element (either a player to render the video, or storage for offline retrieval). In this example, you use the Producer SDK element as a *sink*, or media destination, for your video source (webcam or

IP camera). The plugin element that encapsulates the SDK then sends the video stream to Kinesis Video Streams.

This topic describes how to construct a GStreamer media pipeline that's capable of streaming video from a video source, such as a web camera or RTSP stream, typically connected through intermediate encoding stages (using H.264 encoding) to Kinesis Video Streams. When your video stream is available as a Kinesis video stream, you can use the Kinesis Video Streams Parser Library for further processing, playback, storage, or analysis of your video stream.



Download, build, and configure the GStreamer element

The GStreamer Plugin example is included with the Kinesis Video Streams C++ Producer SDK. For information about SDK prerequisites and downloading, see Step 1: Download and configure the C++ Producer Library code.

You can build the Producer SDK GStreamer sink as a dynamic library on macOS, Ubuntu, Raspberry Pi, or Windows. The GStreamer plugin is located in your `build` directory. To load this plugin, it must be in your `GST_PLUGIN_PATH`. Run the following command:

```
export GST_PLUGIN_PATH=`pwd`/build
```

> **Note**
>
> On macOS, you can only stream video from a network camera when running GStreamer in a Docker container. Streaming video from a USB camera on macOS in a Docker container is not supported.

Run the GStreamer element

To run GStreamer with the Kinesis Video Streams Producer SDK element as a sink, use the `gst-launch-1.0` command. Use upstream elements that are appropriate for the GStreamer plugin to use. For example, v4l2src for v4l2 devices on Linux systems, or rtspsrc for RTSP devices. Specify `kvssink` as the sink (final destination of the pipeline) to send video to the Producer SDK.

In addition to providing credentials and providing a region, the `kvssink`element has the following required parameter:

- `stream-name` – The name of the destination Kinesis Video Streams.

For information about `kvssink` optional parameters, see GStreamer element parameter reference.

For the latest information about GStreamer plugins and parameters, see GStreamer Plugins. You may also use `gst-inspect-1.0` followed by the name of a GStreamer element or plugin to print its information and to verify that it is available on your device:

```
gst-inspect-1.0 kvssink
```

If building `kvssink` failed or GST_PLUGIN_PATH is not properly set, your output will look similar to this:

```
No such element or plugin 'kvssink'
```

Example GStreamer launch commands

The following examples demonstrate how to use the `kvssink` GStreamer plugin to stream video from different types of devices.

Example 1: Stream video from an RTSP camera on Ubuntu

The following command creates a GStreamer pipeline on Ubuntu that streams from a network RTSP camera, using the rtspsrc GStreamer plugin:

```
gst-launch-1.0 -v rtspsrc location="rtsp://YourCameraRtspUrl" short-header=TRUE !
 rtph264depay ! h264parse ! kvssink stream-name="YourStreamName" storage-size=128
```

Example 2: Encode and stream video from a USB camera on Ubuntu

The following command creates a GStreamer pipeline on Ubuntu that encodes the stream from a USB camera in H.264 format, and streams it to Kinesis Video Streams. This example uses the v4l2src GStreamer plugin.

```
gst-launch-1.0 v4l2src do-timestamp=TRUE device=/dev/video0 ! videoconvert ! video/x-
raw,format=I420,width=640,height=480,framerate=30/1 ! x264enc  bframes=0 key-int-max=45
 bitrate=500 ! video/x-h264,stream-format=avc,alignment=au,profile=baseline ! kvssink
 stream-name="YourStreamName" storage-size=512 access-key="YourAccessKey" secret-
key="YourSecretKey" aws-region="YourAWSRegion"
```

Example 3: Stream pre-encoded video from a USB camera on Ubuntu

The following command creates a GStreamer pipeline on Ubuntu that streams video that the camera has already encoded in H.264 format to Kinesis Video Streams. This example uses the v4l2src GStreamer plugin.

```
gst-launch-1.0 v4l2src do-timestamp=TRUE device=/dev/video0 ! h264parse ! video/x-
h264,stream-format=avc,alignment=au ! kvssink stream-name="plugin" storage-size=512
 access-key="YourAccessKey" secret-key="YourSecretKey" aws-region="YourAWSRegion"
```

Example 4: Stream video from a network camera on macOS

The following command creates a GStreamer pipeline on macOS that streams video to Kinesis Video Streams from a network camera. This example uses the rtspsrc GStreamer plugin.

```
gst-launch-1.0 rtspsrc location="rtsp://YourCameraRtspUrl" short-header=TRUE !
 rtph264depay ! h264parse ! video/x-h264, format=avc,alignment=au ! kvssink
 stream-name="YourStreamName" storage-size=512  access-key="YourAccessKey" secret-
key="YourSecretKey" aws-region="YourAWSRegion"
```

Example 5: Stream video from a network camera on Windows

The following command creates a GStreamer pipeline on Windows that streams video to Kinesis Video Streams from a network camera. This example uses the rtspsrc GStreamer plugin.

```
gst-launch-1.0 rtspsrc location="rtsp://YourCameraRtspUrl" short-header=TRUE !
 rtph264depay ! video/x-h264, format=avc,alignment=au ! kvssink stream-
```

```
name="YourStreamName" storage-size=512  access-key="YourAccessKey" secret-
key="YourSecretKey" aws-region="YourAWSRegion"
```

Example 6: Stream video from a camera on Raspberry Pi

The following command creates a GStreamer pipeline on Raspberry Pi that streams video to Kinesis Video Streams. This example uses the v4l2src GStreamer plugin.

```
gst-launch-1.0 v4l2src do-timestamp=TRUE device=/dev/video0 ! videoconvert !
 video/x-raw,format=I420,width=640,height=480,framerate=30/1 !
 omxh264enc control-rate=1 target-bitrate=5120000 periodicity-
idr=45 inline-header=FALSE ! h264parse ! video/x-h264,stream-
format=avc,alignment=au,width=640,height=480,framerate=30/1,profile=baseline ! kvssink
 stream-name="YourStreamName" access-key="YourAccessKey" secret-key="YourSecretKey"
 aws-region="YourAWSRegion"
```

Example 7: Stream both audio and video in Raspberry Pi and Ubuntu

See how to run the gst-launch-1.0 command to start streaming both audio and video in Raspberry-PI and Ubuntu.

Example 8: Stream both audio and video from device sources in macOS

See how to run the gst-launch-1.0 command to start streaming both audio and video in MacOS.

Example 9: Upload MKV file that contains both audio and video

See how to run the gst-launch-1.0 command to upload MKV file that contains both audio and video. You will need an MKV test file with h.264 and AAC encoded media.

Run the GStreamer element in a Docker container

Docker is a platform for developing, deploying, and running applications using containers. Using Docker to create the GStreamer pipeline standardizes the operating environment for Kinesis Video Streams, which streamlines building and using the application.

To install and configure Docker, see the following:

- Docker download instructions

- [Getting started with Docker]

After installing Docker, you can download the Kinesis Video Streams C++ Producer SDK (and GStreamer plugin) from the Amazon Elastic Container Registry using one of the below provided `docker pull` commands.

To run GStreamer with the Kinesis Video Streams Producer SDK element as a sink in a Docker container, do the following:

Topics

Authenticate your Docker client

Authenticate your Docker client to the Amazon ECR registry that you intend to pull your image from. You must get authentication tokens for each registry used. Tokens are valid for 12 hours. For more information, see Registry Authentication in the *Amazon Elastic Container Registry User Guide*.

Example : Authenticate with Amazon ECR

To authenticate with Amazon ECR, copy and paste the following command as is shown.

```
sudo aws ecr get-login-password --region us-west-2 | docker login -u AWS --password-stdin https://546150905175.dkr.ecr.us-west-2.amazonaws.com
```

If successful, the output prints `Login Succeeded`.

Download the Docker image for Ubuntu, macOS, Windows, or Raspberry Pi

Download the Docker image to your Docker environment using one the following commands, depending on your operating system:

Download the Docker image for Ubuntu

```
sudo docker pull 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-cpp-amazon-linux:latest
```

Download the Docker image for macOS

```
docker pull 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-
cpp-amazon-linux:latest
```

Download the Docker image for Windows

```
docker pull 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-
cpp-amazon-windows:latest
```

Download the Docker image for Raspberry Pi

```
sudo docker pull 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-
sdk-cpp-raspberry-pi:latest
```

To verify that the image was successfully added, use the following command:

```
docker images
```

Run the Docker image

Use one of the following commands to run the Docker image, depending on your operating system:

Run the Docker image on Ubuntu

```
sudo docker run -it --network="host" --device=/dev/video0 546150905175.dkr.ecr.us-
west-2.amazonaws.com/kinesis-video-producer-sdk-cpp-amazon-linux /bin/bash
```

Run the Docker image on macOS

```
sudo docker run -it --network="host" 546150905175.dkr.ecr.us-west-2.amazonaws.com/
kinesis-video-producer-sdk-cpp-amazon-linux /bin/bash
```

Run the Docker image on Windows

```
docker run -it 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-
cpp-windows AWS_ACCESS_KEY_ID AWS_SECRET_ACCESS_KEY RTSP_URL STREAM_NAME
```

Run the Docker image on Raspberry Pi

```
sudo docker run -it --device=/dev/video0 --device=/dev/vchiq -v /opt/vc:/opt/vc
 546150905175.dkr.ecr.us-west-2.amazonaws.com/kinesis-video-producer-sdk-cpp-raspberry-
pi /bin/bash
```

Docker launches the container and presents you with a command prompt for using commands within the container.

In the container, set the environment variables using the following command:

```
export LD_LIBRARY_PATH=/opt/awssdk/amazon-kinesis-video-streams-producer-sdk-cpp/
kinesis-video-native-build/downloads/local/lib:$LD_LIBRARY_PATH
export PATH=/opt/awssdk/amazon-kinesis-video-streams-producer-sdk-cpp/kinesis-video-
native-build/downloads/local/bin:$PATH
export GST_PLUGIN_PATH=/opt/awssdk/amazon-kinesis-video-streams-producer-sdk-cpp/
kinesis-video-native-build/downloads/local/lib:$GST_PLUGIN_PATH
```

Start streaming to `kvssink` using the `gst-launch-1.0` to run a pipeline appropriate for your device and video source. For example pipelines, see Example GStreamer launch commands.

GStreamer element parameter reference

To send video to the Amazon Kinesis Video Streams Producer C++ SDK, you specify `kvssink` as the *sink*, or final destination of the pipeline. This reference provides information about `kvssink` required and optional parameters. For more information, see the section called "GStreamer Plugin - kvssink".

Topics

- the section called "Provide credentials to `kvssink`"
- the section called "Provide a region to `kvssink`"
- the section called "`kvssink` optional parameters"

Provide credentials to `kvssink`

To allow the `kvssink` GStreamer element to make requests to AWS, provide AWS credentials for it to use when it calls the Amazon Kinesis Video Streams service. The credential provider chain looks for credentials in the following order:

1. AWS IoT credentials

To set up AWS IoT credentials, see the section called “Controlling access to Kinesis Video Streams resources using AWS IoT”.

The `iot-credentials` parameter value must start with `iot-certificate`, and be followed by a comma-separated list of the following *key*=*value* pairs.

Key	Required	Description
`ca-path`	Yes	File path to the CA certificate used to establish trust with the backend service through TLS. **Example** **Example:** /*file*/*path*/*to*/certificate.pem
`cert-path`	Yes	File path to the X.509 certificate. **Example** **Example:** /*file*/*path*/*to*/*certificateID* -certificate.pem.crt
`endpoint`	Yes	The AWS IoT Core credential endpoint provider endpoint for your AWS account. See the AWS IoT Developer Guide. **Example** **Example:** *credential-account-specific-prefix* .credenti

Key	Required	Description
		als.iot. *aws-region* .amazonaws.com
`key-path`	Yes	File path to the private key used in the public/private key pair. **Example** **Example:** /*file*/*path*/*to*/*certificateID* -private.pem.key
`role-aliases`	Yes	The name of the role alias pointing to the AWS IAM role to use when connecting to AWS IoT Core. **Example** **Example:** *KvsCameraIoTRoleAlias*
`iot-thing-name`	No	The `iot-thing-name` is optional. If `iot-thing-name` is not provided, the `stream-name` parameter value is used. **Example** **Example:** *kvs_example_camera*

Example

Example:

```
gst-launch-1.0 -v ... ! kvssink stream-name="YourStream" aws-region="YourRegion"
 iot-certificate="iot-certificate,endpoint=credential-account-specific-
prefix.credentials.iot.aws-region.amazonaws.com,cert-path=certificateID-
certificate.pem.crt,key-path=certificateID-private.pem.key,ca-
path=certificate.pem,role-aliases=YourRoleAlias,iot-thing-name=YourThingName"
```

2. Environment variables

To have kvssink use credentials from the environment, set the following environment variables:

Environment Variable Name	Required	Description
AWS_ACCESS_KEY_ID	Yes	The AWS access key that's used to access Amazon Kinesis Video Streams.
AWS_SECRET_ACCESS_KEY	Yes	The AWS secret key associated with the access key.
AWS_SESSION_TOKEN	No	Specifies the required session token value if you use temporary security credentials directly from AWS STS operations.

Setting the environment variable changes the value used until the end of your shell session, or until you set the variable to a different value. To make the variables persistent across future sessions, set them in your shell's startup script.

3. access-key, secret-key parameters

To specify credentials directly as a kvssink parameter, set the following parameters:

kvssink Parameter Name	Required	Description
access-key	Yes	The AWS access key that's used to access Amazon Kinesis Video Streams.

kvssink Parameter Name	Required	Description
`secret-key`	Yes	The AWS secret key associate d with the access key.
`session-token`	No	Specifies the required session token value if you use temporary security credentia ls directly from AWS STS operations.

Example

Using static credentials:

```
gst-launch-1.0 -v ... ! kvssink stream-name="YourStream" aws-region="YourRegion"
 access-key="AKIDEXAMPLE" secret-key="SKEXAMPLE"
```

Example

Using temporary credentials:

```
gst-launch-1.0 -v ... ! kvssink stream-name="YourStream" aws-region="YourRegion"
 access-key="AKIDEXAMPLE" secret-key="SKEXAMPLE" session-token="STEXAMPLE"
```

4. Credential file

Important

If you've selected one of the previous methods, you can't use the `credential-file` `kvssink` parameter.

kvssink Parameter Name	Required	Description
`credential-file`	Yes	Path to the text file containin g credentials in a specific format.

The text file must contain credentials in one of the following formats:

- CREDENTIALS *YourAccessKey YourSecretKey*
- CREDENTIALS *YourAccessKey Expiration YourSecretKey SessionToken*

Example

Example: Your *credentials*.txt file is located at /home/ubuntu and contains the following:

CREDENTIALS *AKIDEXAMPLE 2023-08-10T22:43:00Z SKEXAMPLE STEXAMPLE*

To use it in kvssink, type:

```
gst-launch-1.0 -v ... ! kvssink stream-name="YourStream" aws-region="YourRegion"
 credential-file="/home/ubuntu/credentials.txt"
```

Note

The expiration time should be at least 5 + 30 + 3 = **38** seconds in the future. The grace period is defined as the IOT_CREDENTIAL_FETCH_GRACE_PERIOD variable in IotCredentialProvider.h. If the credentials are too close to the expiration when you start kvssink, you receive the error code 0x52000049 - STATUS_INVALID_TOKEN_EXPIRATION.

Important

kvssink doesn't modify the credentials file. If you're using temporary credentials, the credentials file must be updated by an outside source before the expiration time minus the grace period.

Provide a region to kvssink

The following is the region lookup order:

1. AWS_DEFAULT_REGION environment variable is reviewed first. If it is set, that region is used to configure the client.
2. aws-region parameter is reviewed next. If it is set, that region is used to configure the client.

3. If neither of the previous methods were used, kvssink defaults to us-west-2.

kvssink optional parameters

The kvssink element has the following optional parameters. For more information about these parameters, see Kinesis video stream structures.

Parameter	Description	Unit/ Type	Default
stream-name	The name of the destination Amazon Kinesis video stream. **Important** If no stream-name is specified, the default stream name will be used: "DEFAULT_STREAM". If a stream with that default name does not already exist, it will be created.		
absolute-fragment-times	Whether to use absolute fragment times.	Boolean	true
access-key	The AWS access key that's used to		

Parameter	Description	Unit/ Type	Default
	access Kinesis Video Streams. You must either have AWS credentials set or provide this parameter. To provide this information, type the following: `export AWS_ACCESS_KEY_ID=`		
`avg-bandwidth-bps`	The expected average bandwidth for the stream.	Bits per second	4194304

Parameter	Description	Unit/ Type	Default
`aws-region`	The AWS Region to use. **Note** You can also provide the region with the `AWS_DEFAULT_REGION` environment variable. The environment variables take precedence if both the environment variable and kvssink parameters are set. **Important** The region will default to `us-west-2` if not otherwise specified.	String	`"us-west-2"`

Parameter	Description	Unit/ Type	Default
`buffer-duration`	The stream buffer duration.	Seconds	120
`codec-id`	The codec ID of the stream.	String	`"V_MPEG4/ISO/AVC"`
`connection-staleness`	The time after, which the stream staleness callback is called.	Seconds	60
`content-type`	The content type of the stream.	String	`"video/h264"`
`fragment-acks`	Whether to use fragment ACKs.	Boolean	true
`fragment-duration`	The fragment duration that you want.	Milliseconds	2000
`framerate`	The expected frame rate.	Frames per second	25
`frame-timecodes`	Whether to use frame timecodes or generate timestamp s using the current time callback.	Boolean	true
`key-frame-fragmentation`	Whether to produce fragments on a key frame.	Boolean	true
`log-config`	The log configuration path.	String	`"../kvs_log_configuration"`

Parameter	Description	Unit/ Type	Default
`max-latency`	The maximum latency for the stream.	Seconds	60
`recalculate-metrics`	Whether to recalculate the metrics.	Boolean	true
`replay-duration`	The duration to roll the current reader backward to replay during an error if restarting is enabled.	Seconds	40
`restart-on-error`	Whether to restart when an error occurs.	Boolean	true
`retention-period`	The length of time the stream is preserved.	Hours	2
`rotation-period`	The key rotation period. For more information, see [Rotating AWS KMS Keys]().	Seconds	3600

Parameter	Description	Unit/ Type	Default
`secret-key`	The AWS secret key that's used to access Kinesis Video Streams. You must either have AWS credentials set or provide this parameter. `export AWS_SECRET_ACCESS_KEY=`		
`session-token`	Specifies the required session token value if you use temporary security credentials directly from AWS STS operations.		
`storage-size`	The device storage size in mebibyte (MiB). For information about configuring device storage, see StorageInfo.	Mebibyte (MiB)	128
`streaming-type`	The streaming type. Valid values include: • 0: real time • 1: near real time (not currently supported) • 2: offline	Enum `GstKvsSinkStreamingType`	0: real time

Parameter	Description	Unit/ Type	Default
`timecode-scale`	The MKV timecode scale.	Milliseconds	1
`track-name`	The MKV track name.	String	`"kinesis_video"`

Parameter	Description	Unit/ Type	Default
`iot-certificate`	AWS IoT credentials to be used in the `kvssink` element. `iot-certificate` accepts the following keys and values: **Note** The `iot-thing-name` is **optional**. If `iot-thing-name` is not provided, the `stream-name` parameter value is used. • `endpoint=iotcredentialsprov iderendpoint` • `cert-path =/localdir ectorypat h /to/certi ficate` • `key-path=/ localdir ectorypath /`	String	None

Parameter	Description	Unit/ Type	Default
	`to/private/key` • `ca-path=/localdirectorypath/to/ca-cert` • `role-aliases =role-aliases` • `iot-thing-name=YourIotThingName`		

Example: Sending data to Kinesis Video Streams using the PutMedia API

This example demonstrates how to use the PutMedia API. It shows how to send data that's already in a container format (MKV). If your data must be assembled into a container format before sending (for example, if you are assembling camera video data into frames), see Kinesis Video Streams Producer Libraries.

Note

The `PutMedia` operation is available only in the C++ and Java SDKs. This is due to the full-duplex management of connections, data flow, and acknowledgements. It's not supported in other languages.

This example includes the following steps:

- Step 1: Download and configure the code
- Step 2: Write and examine the code
- Step 3: Run and verify the code

Step 1: Download and configure the code

Follow the steps to download the Java example code, import the project into your Java IDE, configure the library locations, and configure the code to use your AWS credentials.

1. Create a directory and clone the example source code from the GitHub repository. The `PutMedia` example is part of the Java Producer Library.

```
git clone https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-java
```

2. Open the Java IDE that you're using (for example, Eclipse or IntelliJ IDEA), and import the Apache Maven project that you downloaded:

   - **In Eclipse:** Choose **File**, **Import**, **Maven**, **Existing Maven Projects**, and navigate to the root of the downloaded package. Select the `pom.xml` file.
   - **In IntelliJ Idea:** Choose **Import**. Navigate to the `pom.xml` file in the root of the downloaded package.

   For more information, see the related IDE documentation.

3. Update the project so that the IDE can find the libraries that you imported.

   - For IntelliJ IDEA, do the following:

     a. Open the context (right-click) menu for the project's **lib** directory, and choose **Add as library**.

     b. Choose **File**, then choose**Project Structure**.

     c. Under **Project Settings**, choose **Modules**.

     d. In the **Sources** tab, set **Language Level** to **7** or higher.

   - For Eclipse, do the following:

     a. Open the context (right-click) menu for the project, and choose **Properties**, **Java Build Path**, **Source**. Then do the following:

        1. On the **Source** tab, double-click **Native library location**.
        2. In the **Native Library Folder Configuration** wizard, choose **Workspace**.
        3. In the **Native Library Folder** selection, choose the **lib** directory in the project.

b. Open the context (right-click) menu for the project, and choose **Properties**. Then do the following:

   1. On the **Libraries** tab, choose **Add Jars**.
   2. In the **JAR selection** wizard, choose all the .jars in the project's `lib` directory.

Step 2: Write and examine the code

The `PutMedia` API example (`PutMediaDemo`) shows the following coding pattern:

Topics

- Create the PutMediaClient
- Stream media and pause the thread

The code examples in this section are from the `PutMediaDemo` class.

Create the PutMediaClient

Creating the `PutMediaClient` object requires the following parameters:

- The URI for the `PutMedia` endpoint.
- An `InputStream` pointing to the MKV file to stream.
- The stream name. This example uses the stream that was created in the Using the Java Producer Library (`my-stream`). To use a different stream, change the following parameter:

```
private static final String STREAM_NAME="my-stream";
```

> **Note**
>
> The `PutMedia` API example doesn't create a stream. You must create a stream either by using the test application for the Using the Java Producer Library, the Kinesis Video Streams console, or the AWS CLI.

- The current timestamp.
- The time code type. The example uses `RELATIVE`, indicating that the timestamp is relative to the start of the container.

- An `AWSKinesisVideoV4Signer` object that verifies that the received packets were sent by the authorized sender.
- The maximum upstream bandwidth in Kbps.
- An `AckConsumer` object to receive packet received acknowledgements.

The following code creates the `PutMediaClient` object:

```
/* actually URI to send PutMedia request */
final URI uri = URI.create(KINESIS_VIDEO_DATA_ENDPOINT + PUT_MEDIA_API);

/* input stream for sample MKV file */
final InputStream inputStream = new FileInputStream(MKV_FILE_PATH);

/* use a latch for main thread to wait for response to complete */
final CountDownLatch latch = new CountDownLatch(1);

/* a consumer for PutMedia ACK events */
final AckConsumer ackConsumer = new AckConsumer(latch);

/* client configuration used for AWS SigV4 signer */
final ClientConfiguration configuration = getClientConfiguration(uri);

/* PutMedia client */
final PutMediaClient client = PutMediaClient.builder()
        .putMediaDestinationUri(uri)
        .mkvStream(inputStream)
        .streamName(STREAM_NAME)
        .timestamp(System.currentTimeMillis())
        .fragmentTimeCodeType("RELATIVE")
        .signWith(getKinesisVideoSigner(configuration))
        .upstreamKbps(MAX_BANDWIDTH_KBPS)
        .receiveAcks(ackConsumer)
        .build();
```

Stream media and pause the thread

After the client is created, the sample starts asynchronous streaming with `putMediaInBackground`. The main thread is then paused with `latch.await` until the `AckConsumer` returns, at which point the client is closed.

```
 /* start streaming video in a background thread */
```

```
            client.putMediaInBackground();

            /* wait for request/response to complete */
            latch.await();

            /* close the client */
            client.close();
```

Step 3: Run and verify the code

To run the `PutMedia` API example, do the following:

1. Create a stream named `my-stream` in the Kinesis Video Streams console or by using the AWS CLI.
2. Change your working directory to the Java producer SDK directory:

   ```
   cd /<YOUR_FOLDER_PATH_WHERE_SDK_IS_DOWNLOADED>/amazon-kinesis-video-streams-producer-sdk-java/
   ```

3. Compile the Java SDK and demo application:

   ```
   mvn package
   ```

4. Create a temporary filename in the `/tmp` directory:

   ```
   jar_files=$(mktemp)
   ```

5. Create a classpath string of dependencies from the local repository to a file:

   ```
   mvn -Dmdep.outputFile=$jar_files dependency:build-classpath
   ```

6. Set the value of the `LD_LIBRARY_PATH` environment variable as follows:

   ```
   export LD_LIBRARY_PATH=/<YOUR_FOLDER_PATH_WHERE_SDK_IS_DOWNLOADED>/amazon-kinesis-video-streams-producer-sdk-cpp/kinesis-video-native-build/downloads/local/lib:$LD_LIBRARY_PATH
   $ classpath_values=$(cat $jar_files)
   ```

7. Run the demo from the command line as follows, providing your AWS credentials:

```
java -classpath target/kinesisvideo-java-demo-1.0-SNAPSHOT.jar:$classpath_values -
Daws.accessKeyId=${ACCESS_KEY} -Daws.secretKey=${SECRET_KEY} -Djava.library.path=/
opt/amazon-kinesis-video-streams-producer-sdk-cpp/kinesis-video-native-build
 com.amazonaws.kinesisvideo.demoapp.DemoAppMain
```

8. Open the Kinesis Video Streams console, and choose your stream on the **Manage Streams** page. The video plays in the **Video Preview** pane.

Example: Streaming from an RTSP source

The C++ Producer Library contains a definition for a Docker container that connects to a Real-Time Streaming Protocol (RTSP) network camera. Using Docker standardizes the operating environment for Kinesis Video Streams, which streamlines building and using the application.

The following procedure demonstrates how to set up and use the RTSP demo application.

Topics

- Video tutorials
- Prerequisites
- Build the Docker image
- Run the RTSP example application

Video tutorials

This video shows how to set up a Raspberry Pi to send RTSP feeds to AWS cloud and Amazon Kinesis Video Streams. This is an end-to-end demonstration.

This video demonstrates how to capture images from a feed to use computer vision and Amazon Rekognition to process the images and send alerts.

Prerequisites

To run the Kinesis Video Streams RTSP example application, you must have the following:

- **Docker:** For information about installing and using Docker, see the following links:
  - Docker download instructions
  - Getting started with Docker

- **RTSP network camera source:** For information about recommended cameras, see System Requirements.

Build the Docker image

First, build the Docker image that the demo application will run inside.

1. Clone the Amazon Kinesis Video Streams demos repository.

```
git clone https://github.com/aws-samples/amazon-kinesis-video-streams-demos.git
```

2. Change to the directory containing the Dockerfile. In this case, it is the docker-rtsp directory.

```
cd amazon-kinesis-video-streams-demos/producer-cpp/docker-rtsp/
```

3. Use the following command to build the Docker image. This command creates the image and tags it as rtspdockertest.

```
docker build -t rtspdockertest .
```

4. Run `docker images` and search for the image ID tagged with `rtspdockertest`.

   For example, in the sample output below, the `IMAGE ID` is `54f0d65f69b2`.

```
REPOSITORY        TAG       IMAGE ID        CREATED            PLATFORM       SIZE
       BLOB SIZE
rtspdockertest    latest    54f0d65f69b2    10 minutes ago     linux/arm64    653.1
 MiB    292.4 MiB
```

   You will need this in a later step.

Run the RTSP example application

You can run the RTSP example application either from within or outside the Docker container. Follow the appropriate instructions below.

Topics

- Within the Docker container
- Outside the Docker container

Within the Docker container

Run the RTSP example application

1. Start the Amazon Kinesis Video Streams Docker container using the following command:

```
docker run -it YourImageId /bin/bash
```

2. To start the sample application, provide your AWS credentials, the name of the Amazon Kinesis video stream, and the URL of the RTSP network camera.

> **Important**
>
> If you are using temporary credentials, you'll also need to provide your AWS_SESSION_TOKEN. See the second example below.

```
export AWS_ACCESS_KEY_ID=YourAccessKeyId
export AWS_SECRET_ACCESS_KEY=YourSecretKeyId
export AWS_DEFAULT_REGION=YourAWSRegion
./kvs_gstreamer_sample YourStreamName YourRtspUrl
```

Temporary credentials:

```
export AWS_ACCESS_KEY_ID=YourAccessKeyId
export AWS_SECRET_ACCESS_KEY=YourSecretKeyId
export AWS_SESSION_TOKEN=YourSessionToken
export AWS_DEFAULT_REGION=YourAWSRegion
./kvs_gstreamer_sample YourStreamName YourRtspUrl
```

3. Sign into the AWS Management Console and open the [Kinesis Video Streams console].

   View the stream.

4. To exit the Docker container, close the terminal window or type `exit`.

Outside the Docker container

From **outside** the Docker container, use the following command:

```
docker run -it YourImageId /bin/bash -c "export AWS_ACCESS_KEY_ID=YourAccessKeyId;
 export AWS_SECRET_ACCESS_KEY=YourSecretKeyId; export
 AWS_SESSION_TOKEN=YourSessionToken; export AWS_DEFAULT_REGION=YourAWSRegion; ./
kvs_gstreamer_sample YourStreamName YourRtspUrl"
```

Example: Parsing and rendering Kinesis Video Streams fragments

The Stream Parser Library contains a demo application named `KinesisVideoRendererExample` that demonstrates parsing and rendering Amazon Kinesis video stream fragments. The example uses JCodec to decode the H.264 encoded frames that are ingested using the Example: Kinesis Video Streams Producer SDK GStreamer Plugin - kvssink application. After the frame is decoded using JCodec, the visible image is rendered using JFrame.

This example shows how to do the following:

- Retrieve frames from a Kinesis video stream using the `GetMedia` API and render the stream for viewing.
- View the video content of streams in a custom application instead of using the Kinesis Video Streams console.

You can also use the classes in this example to view Kinesis video stream content that isn't encoded as H.264, such as a stream of JPEG files that don't require decoding before being displayed.

The following procedure demonstrates how to set up and use the Renderer demo application.

Prerequisites

To examine and use the Renderer example library, you must have the following:

- An Amazon Web Services (AWS) account. If you don't already have an AWS account, see Getting Started with Kinesis Video Streams.
- A Java integrated development environment (IDE), such as Eclipse Java Neon or JetBrains IntelliJ Idea.

Running the renderer example

1. Create a directory, and then clone the example source code from the GitHub repository.

```
git clone https://github.com/aws/amazon-kinesis-video-streams-parser-library
```

2. Open the Java IDE that you are using (for example, Eclipse or IntelliJ IDEA), and import the Apache Maven project that you downloaded:

   - **In Eclipse:** Choose **File**, **Import**, **Maven**, **Existing Maven Projects**. Navigate to the `kinesis-video-streams-parser-lib` directory.
   - **In IntelliJ Idea:** Choose **Import**. Navigate to the `pom.xml` file in the root of the downloaded package.

   > **Note**
   >
   > If IntelliJ can't find your dependencies, you might have to do the following:
   >
   > - **Build clean:** Choose **File**, **Settings**, **Build, Execution, Deployment**, **Compiler**. Verify that **Clear output directory on rebuild** is selected, and then choose **Build**, **Build Project**.
   > - **Reimport the project:** Open the context (right-click) menu for the project, and choose **Maven**, **Reimport**.

   For more information, see the related IDE documentation.

3. From your Java IDE, open `src/test/java/com.amazonaws.kinesisvideo.parser/examples/KinesisVideoRendererExampleTest`.
4. Remove the `@Ignore` directive from the file.
5. Update the `.stream` parameter with the name of your Kinesis video stream.
6. Run the `KinesisVideoRendererExample` test.

How It Works

The example application demonstrates the following:

- Sending MKV data

- Parsing MKV fragments into frames
- Decoding and displaying the frame

Sending MKV data

The example sends sample MKV data from the `rendering_example_video.mkv` file, using `PutMedia` to send video data to a stream named **render-example-stream**.

The application creates a `PutMediaWorker`:

```
PutMediaWorker putMediaWorker = PutMediaWorker.create(getRegion(),
    getCredentialsProvider(),
    getStreamName(),
    inputStream,
    streamOps.amazonKinesisVideo);
executorService.submit(putMediaWorker);
```

For information about the `PutMediaWorker` class, see Call PutMedia in the Stream Parser Library documentation.

Parsing MKV fragments into frames

The example then retrieves and parses the MKV fragments from the stream using a `GetMediaWorker`:

```
GetMediaWorker getMediaWorker = GetMediaWorker.create(getRegion(),
    getCredentialsProvider(),
    getStreamName(),
    new StartSelector().withStartSelectorType(StartSelectorType.EARLIEST),
    streamOps.amazonKinesisVideo,
    getMediaProcessingArgumentsLocal.getFrameVisitor());
executorService.submit(getMediaWorker);
```

For more information about the `GetMediaWorker` class, see Call GetMedia in the Stream Parser Library documentation.

Decoding and displaying the frame

The example then decodes and displays the frame using JFrame.

The following code example is from the `KinesisVideoFrameViewer` class, which extends `JFrame`:

```
 public void setImage(BufferedImage bufferedImage) {
    image = bufferedImage;
    repaint();
}
```

The image is displayed as an instance of java.awt.image.BufferedImage. For examples that show how to work with `BufferedImage`, see Reading/Loading an Image.

Monitoring Amazon Kinesis Video Streams

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon Kinesis Video Streams and your AWS solutions. We recommend collecting monitoring data from all of the parts of your AWS solution to help you debug a multi-point failure if one occurs. Before you start monitoring Amazon Kinesis Video Streams, we recommend that you create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- What resources will you monitor?
- How often will you monitor these resources?
- What monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

After you've defined your monitoring goals and created your monitoring plan, the next step is to establish a baseline for normal Amazon Kinesis Video Streams performance in your environment. You should measure Amazon Kinesis Video Streams performance at various times and under different load conditions. As you monitor Amazon Kinesis Video Streams, store a history of monitoring data that you've collected. You can compare current Amazon Kinesis Video Streams performance to this historical data to help you identify normal performance patterns and performance anomalies, and devise methods to address issues that might arise.

Topics

Monitoring Amazon Kinesis Video Streams metrics with CloudWatch

You can monitor a Kinesis video stream using Amazon CloudWatch, which collects and processes raw data from Amazon Kinesis Video Streams into readable, near real-time metrics. These statistics are recorded for a period of 15 months so that you can access historical information and gain a better perspective on how your web application or service is performing.

In the Amazon Kinesis Video Streams console, you can view CloudWatch metrics for a Amazon Kinesis video stream in two ways:

- In the **Dashboard** page, choose the **Video streams** tab in the **Account-level metrics for Current Region** section.
- Choose the **Monitoring** tab in the video stream's details page.

Amazon Kinesis Video Streams provides the following metrics:

Metric	Description
`ArchivedFragmentsConsumed.Media`	The number of fragment media quota points that were consumed by all of the APIs. For an explanation of the concept of quota points, see the section called “Fragment-metadata and fragment-media quotas”. Units: Count
`ArchivedFragmentsConsumed.Metadata`	The number of fragments metadata quota points that were consumed by all of the APIs. For an explanation of the concept of quota points, see the section called “Fragment-metadata and fragment-media quotas”. Units: Count
`PutMedia.Requests`	The number of `PutMedia` API requests for a given stream. Units: Count
`PutMedia.IncomingBytes`	The number of bytes received as part of `PutMedia` for the stream. Units: Bytes

Metric	Description
PutMedia.IncomingFragments	The number of complete fragments received as part of PutMedia for the stream. Units: Count
PutMedia.IncomingFrames	The number of complete frames received as part of PutMedia for the stream. Units: Count
PutMedia.ActiveConnections	The total number of connections to the service host. Units: Count
PutMedia.ConnectionErrors	The errors while establishing PutMedia connection for the stream. Units: Count
PutMedia.FragmentIngestionLatency	The time difference between when the first and last bytes of a fragment are received by Amazon Kinesis Video Streams. Units: Milliseconds
PutMedia.FragmentPersistLatency	The time taken from when the complete fragment data is received and archived. Units: Count
PutMedia.Latency	The time difference between the request and the HTTP response from InletService while establishing the connection. Units: Count

Metric	Description
`PutMedia.BufferingAckLatency`	The time difference between when the first byte of a new fragment is received by Amazon Kinesis Video Streams and when the Buffering ACK is sent for the fragment. Units: Milliseconds
`PutMedia.ReceivedAckLatency`	The time difference between when the last byte of a new fragment is received by Amazon Kinesis Video Streams and when the Received ACK is sent for the fragment. Units: Milliseconds
`PutMedia.PersistedAckLatency`	The time difference between when the last byte of a new fragment is received by Amazon Kinesis Video Streams and when the Persisted ACK is sent for the fragment. Units: Milliseconds
`PutMedia.ErrorAckCount`	The number of Error ACKs sent while doing `PutMedia` for the stream. Units: Count
`PutMedia.Success`	1 for each fragment successfully written; 0 for every failed fragment. The average value of this metric indicates how many complete, valid fragments are sent. Units: Count
`GetMedia.Requests`	The number of `GetMedia` API requests for a given stream. Units: Count

Metric	Description
GetMedia.OutgoingBytes	The total number of bytes sent out from the service as part of the GetMedia API for a given stream. Units: Bytes
GetMedia.OutgoingFragments	The number of fragments sent while doing GetMedia for the stream. Units: Count
GetMedia.OutgoingFrames	The number of frames sent during GetMedia on the given stream. Units: Count
GetMedia.MillisBehindNow	The time difference between the current server timestamp and the server timestamp of the last fragment sent. Units: Milliseconds
GetMedia.ConnectionErrors	The number of connections that were not successfu lly established. Units: Count

Metric	Description
GetMedia.Success	1 for every fragment successfully sent; 0 for every failure. The average value indicates the rate of success. **Note** Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging. Units: Count
GetMediaForFragmentList.OutgoingBytes	The total number of bytes sent out from the service as part of the GetMediaForFragmentList API for a given stream. Units: Bytes
GetMediaForFragmentList.OutgoingFragments	The total number of fragments sent out from the service as part of the GetMediaForFragmentList API for a given stream. Units: Count
GetMediaForFragmentList.OutgoingFrames	The total number of frames sent out from the service as part of the GetMediaForFragmentList API for a given stream. Units: Count
GetMediaForFragmentList.Requests	The number of GetMediaForFragmentList API requests for a given stream. Units: Count

Metric	Description
GetMediaForFragmentList.Success	1 for every fragment successfully sent; 0 for every failure. The average value indicates the rate of success. **Note** Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging. Units: Count
ListFragments.Latency	The latency of the ListFragments API calls for the given stream name. Units: Milliseconds
ListFragments.Requests	The number of ListFragments API requests for a given stream. Units: Count

Metric	Description
`ListFragments.Success`	1 for every successful request; 0 for every failure. The average value indicates the rate of success. **Note** Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging. Units: Count
`GetHLSStreamingSessionURL.Latency`	The latency of the `GetHLSStreamingSessionURL` API calls for the given stream name. Units: Milliseconds
`GetHLSStreamingSessionURL.Requests`	The number of `GetHLSStreamingSessionURL` API requests for a given stream. Units: Count

Metric	Description
`GetHLSStreamingSessionURL.Success`	1 for every successful request; 0 for every failure. The average value indicates the rate of success. **Note** Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging. Units: Count
`GetHLSMasterPlaylist.Latency`	The latency of the `GetHLSMasterPlaylist` API calls for the given stream name. Units: Milliseconds
`GetHLSMasterPlaylist.Requests`	The number of `GetHLSMasterPlaylist` API requests for a given stream. Units: Count

Metric	Description
`GetHLSMasterPlaylist.Success`	1 for every successful request; 0 for every failure. The average value indicates the rate of success. **Note** Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging. Units: Count
`GetHLSMediaPlaylist.Latency`	The latency of the `GetHLSMediaPlaylist` API calls for the given stream name. Units: Milliseconds
`GetHLSMediaPlaylist.Requests`	The number of `GetHLSMediaPlaylist` API requests for a given stream. Units: Count

Metric	Description
`GetHLSMediaPlaylist.Success`	1 for every successful request; 0 for every failure. The average value indicates the rate of success. **Note** Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging. Units: Count
`GetMP4InitFragment.Latency`	The latency of the `GetMP4InitFragment` API calls for the given stream name. Units: Milliseconds
`GetMP4InitFragment.Requests`	The number of `GetMP4InitFragment` API requests for a given stream. Units: Count

<table>
<tr><th>Metric</th><th>Description</th></tr>
<tr><td><code>GetMP4InitFragment.Success</code></td><td>1 for every successful request; 0 for every failure. The average value indicates the rate of success.

Note
Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging.

Units: Count</td></tr>
<tr><td><code>GetMP4MediaFragment.Latency</code></td><td>The latency of the <code>GetMP4MediaFragment</code> API calls for the given stream name.

Units: Milliseconds</td></tr>
<tr><td><code>GetMP4MediaFragment.Requests</code></td><td>The number of <code>GetMP4MediaFragment</code> API requests for a given stream.

Units: Count</td></tr>
</table>

Metric	Description
GetMP4MediaFragment.Success	1 for every successful request; 0 for every failure. The average value indicates the rate of success. **Note** Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging. Units: Count
GetMP4MediaFragment.OutgoingBytes	The total number of bytes sent out from the service as part of the GetMP4MediaFragment API for a given stream. Units: Bytes
GetTSFragment.Latency	The latency of the GetTSFragment API calls for the given stream name. Units: Milliseconds
GetTSFragment.Requests	The number of GetTSFragment API requests for a given stream. Units: Count

Metric	Description
`GetTSFragment.Success`	1 for every successful request; 0 for every failure. The average value indicates the rate of success. **Note** Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging. Units: Count
`GetTSFragment.OutgoingBytes`	The total number of bytes sent out from the service as part of the `GetTSFragment` API for a given stream. Units: Bytes
`GetDASHStreamingSessionURL.Latency`	The latency of the `GetDASHStreamingSessionURL` API calls for the given stream name. Units: Milliseconds
`GetDASHStreamingSessionURL.Requests`	The number of `GetDASHStreamingSessionURL` API requests for a given stream. Units: Count

Metric	Description
`GetDASHStreamingSessionURL.Success`	1 for every successful request; 0 for every failure. The average value indicates the rate of success. **Note** Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging. Units: Count
`GetDASHManifest.Latency`	The latency of the `GetDASHManifest` API calls for the given stream name. Units: Milliseconds
`GetDASHManifest.Requests`	The number of `GetDASHManifest` API requests for a given stream. Units: Count

Metric	Description
`GetDASHManifest.Success`	1 for every successful request; 0 for every failure. The average value indicates the rate of success. **Note** Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging. Units: Count
`GetClip.Latency`	The latency of the GetClip API calls for the given video stream name. Units: Milliseconds
`GetClip.Requests`	The number of GetClip API requests for a given video stream. Units: Count

Metric	Description
GetClip.Success	1 for every successful request; 0 for every failure. The average value indicates the rate of success. **Note** Failures include both 400 (user) errors and 500 (system) errors. For more information about enabling a summary of requests and responses, including AWS request IDs, see Request/Response Summary Logging. Units: Count
GetClip.OutgoingBytes	The total number of bytes sent out from the service as part of the GetClip API for a given video stream. Units: Bytes

CloudWatch metrics guidance

CloudWatch metrics can help find answers to the following questions:

Topics

Is data reaching the Amazon Kinesis Video Streams service?

Relevant metrics:

- `PutMedia.IncomingBytes`
- `PutMedia.IncomingFragments`
- `PutMedia.IncomingFrames`

Action items:

- If there's a drop in these metrics, check if your application is still sending data to the service.
- Check the network bandwidth. If your network bandwidth is insufficient, it could be slowing down the rate the service is receiving the data.

Why is data not being successfully ingested by the Amazon Kinesis Video Streams service?

Relevant metrics:

- `PutMedia.Requests`
- `PutMedia.ConnectionErrors`
- `PutMedia.Success`
- `PutMedia.ErrorAckCount`

Action items:

- If there's an increase in `PutMedia.ConnectionErrors`, look at the HTTP response and error codes received by the producer client to see what errors are occurring while establishing the connection.
- If there's a drop in `PutMedia.Success` or increase in `PutMedia.ErrorAckCount`, look at the ack error code in the ack responses sent by the service to see why ingestion of data is failing. For more information, see AckErrorCode.Values.

Why can't the data be read from the Amazon Kinesis Video Streams service at the same rate as it's being sent from the producer?

Relevant metrics:

- `PutMedia.FragmentIngestionLatency`
- `PutMedia.IncomingBytes`

Action items:

- If there's a drop in these metrics, check the network bandwidth of your connections. Low-bandwidth connections could cause the data to reach the service at a lower rate.

Why is there no video in the console, or why is the video being played with a delay?

Relevant metrics:

- `PutMedia.FragmentIngestionLatency`
- `PutMedia.FragmentPersistLatency`
- `PutMedia.Success`
- `ListFragments.Latency`
- `PutMedia.IncomingFragments`

Action items:

- If there's an increase in `PutMedia.FragmentIngestionLatency` or a drop in `PutMedia.IncomingFragments`, check the network bandwidth and whether the data is still being sent.
- If there's a drop in `PutMedia.Success`, check the ack error codes. For more information, see AckErrorCode.Values.
- If there's an increase in `PutMedia.FragmentPersistLatency` or `ListFragments.Latency`, you're most likely experiencing a service issue. If the condition persists for an extended period of time, check with your customer service contact to see if there's an issue with your service.

What is the delay in reading real-time data, and why is the client lagging behind the head of the stream?

Relevant metrics:

- `GetMedia.MillisBehindNow`
- `GetMedia.ConnectionErrors`
- `GetMedia.Success`

Action items:

- If there's an increase in `GetMedia.ConnectionErrors`, then the consumer might be falling behind in reading the stream, due to frequent attempts to re-connect to the stream. Look at the HTTP response/error codes returned for the `GetMedia` request.
- If there's a drop in `GetMedia.Success`, it’s likely due to the service being unable to send the data to the consumer, which would result in dropped connection, and reconnects from consumers, which would result in the consumer lagging behind the head of the stream.
- If there's an increase in `GetMedia.MillisBehindNow`, look at your bandwidth limits to see if you're receiving the data at a slower rate because of lower bandwidth.

Is the client reading data out of the Kinesis video stream, and at what rate?

Relevant metrics:

- `GetMedia.OutgoingBytes`
- `GetMedia.OutgoingFragments`
- `GetMedia.OutgoingFrames`
- `GetMediaForFragmentList.OutgoingBytes`
- `GetMediaForFragmentList.OutgoingFragments`
- `GetMediaForFragmentList.OutgoingFrames`

Action items:

- These metrics indicate the rate at which real-time and archived data is being read.

Why can't the client read data out of the Kinesis video stream?

Relevant metrics:

- `GetMedia.ConnectionErrors`
- `GetMedia.Success`
- `GetMediaForFragmentList.Success`
- `PutMedia.IncomingBytes`

Action items:

- If there's an increase in `GetMedia.ConnectionErrors`, look at the HTTP response and error codes returned by the `GetMedia` request. For more information, see AckErrorCode.Values.
- If you're trying to read the latest or live data, check `PutMedia.IncomingBytes` to see if there's data coming into the stream for the service to send to the consumers.
- If there's a drop in `GetMedia.Success` or `GetMediaForFragmentList.Success`, it’s likely due to the service being unable to send the data to the consumer. If the condition persists for an extended period of time, check with your customer service contact to see if there's an issue with your service.

Monitoring the Amazon Kinesis Video Streams Edge Agent with CloudWatch

You can monitor the Amazon Kinesis Video Streams Edge Agent using Amazon CloudWatch, which collects and processes raw data into readable, near real-time metrics. These statistics are recorded for a period of 15 months. With this historical information, you can gain a better perspective on how your web application or Amazon Kinesis Video Streams Edge Agent service is performing.

To view the metrics, do the following:

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation, under **Metrics**, select **All Metrics**.
3. Choose the **Browse** tab, then select the **EdgeRuntimeAgent** custom namespace.

Amazon Kinesis Video Streams Edge Agent publishes the following metrics under the namespace `EdgeRuntimeAgent`:

Dimensions	State	Description
Stream name, `RecordJob`	Running	Publishes continuously when the `RecordJob` is running. Units: None. "1" is published for as long as `RecordJob` is in this state.
	FatalError	Publishes if a `RecordJob` fatally errors. Units: None. "1" is published once, when this event occurs. **Note** See logs for additional information.
	Completed	Publishes when a `RecordJob` is completed. Units: None. "1" is published once, when this event occurs.
Stream name, `UploadJob`	Running	Publishes continuously when the `UploadJob` is running. Units: None. "1" is published for as long as `UploadJob` is in this state.
	FatalError	Publishes if the `UploadJob` fatally errors. Units: None. "1" is published once, when this event occurs. **Note** See logs for additional information.
	Completed	Publishes when the `UploadJob` is completed. Units: None. "1" is published once, when this event occurs.

Dimensions	State	Description
Stream name	PercentageSpaceUsed	This is the percentage used out of the total space allocated in Amazon Kinesis Video Streams Edge Agent configurations for recording media. See the section called “LocalSizeConfig” for more information. Units: Percentage (scale 0–1).
Thing name	Alive	Publishes every minute from the Amazon Kinesis Video Streams Edge Agent, regardless of any configurations running on it. This can be used to understand if the Amazon Kinesis Video Streams Edge Agent is alive and ready to accept configurations. Units: None. "1" is published every minute.
	RecordJobs.HealthyJobCount	Total count of running and scheduled record jobs on Amazon Kinesis Video Streams Edge Agent. Units: Count.
	UploadJobs.HealthyJobCount	Total count of running and scheduled upload jobs on Amazon Kinesis Video Streams Edge Agent. Units: Count.
	RecordJobs.UnhealthyJobCount	Total count of currently errored record jobs. Units: Count.
	UploadJobs.UnhealthyJobCount	Total count of currently errored upload jobs. Units: Count.

Dimensions	State	Description
	RecordJobs.RunningJobCount	Total count of actively running record jobs. Units: Count.
	UploadJobs.RunningJobCount	Total count of actively running upload jobs. Units: Count.
	RecordJobs.EdgeConfigCount	Total count of record configurations in process on Amazon Kinesis Video Streams Edge Agent. Units: Count.
	UploadJobs.EdgeConfigCount	Total count of upload configurations in process on Amazon Kinesis Video Streams Edge Agent. Units: Count.

CloudWatch metrics guidance for Amazon Kinesis Video Streams Edge Agent

CloudWatch metrics can be useful for finding answers to the following questions:

Topics

- Does the Amazon Kinesis Video Streams Edge Agent have enough space to record?
- Is the Amazon Kinesis Video Streams Edge Agent alive?
- Are there any unhealthy jobs?
- Do any jobs need external intervention?

Does the Amazon Kinesis Video Streams Edge Agent have enough space to record?

Relevant metrics: `PercentageSpaceUsed`

Action: No action required.

Is the Amazon Kinesis Video Streams Edge Agent alive?

Relevant metrics: `Alive`

Action: If at any point you stop receiving this metric, it means that the Amazon Kinesis Video Streams Edge Agent encountered **one or more** of the following:

- An application runtime issue: memory or other resource constraint, bug, and so on
- The AWS IoT device that the agent is running on shutdown, crashed, or terminated
- The AWS IoT device doesn't have network connectivity

Are there any unhealthy jobs?

Relevant metrics:

- `RecordJobs.UnhealthyJobCount`
- `UploadJobs.UnhealthyJobCount`

Action: Inspect the logs and look for the `FatalError` metric.

- If the `FatalError` metric **is** present, a fatal error was encountered and you need to manually restart the job. Inspect the logs and fix the issue before using `StartEdgeConfigurationUpdate` to manually restart the job.
- If the `FatalError` metric **isn't** present, a transient (non-fatal) error was encountered and Amazon Kinesis Video Streams Edge Agent is retrying the job.

> **Note**
>
> To have the agent reattempt a fatally-errored job, use the section called “StartEdgeConfigurationUpdate”.

Do any jobs need external intervention?

Relevant metrics:

- `PercentageSpaceUsed` – If this exceeds a certain value, the record job is paused and resumes only when space is available (when media goes out of retention). You can send an updated configuration with a higher `MaxLocalMediaSizeInMB` to update the job immediately.
- `RecordJob.FatalError` / `UploadJob.FatalError` – Investigate the agent's logs and send the configuration again for the job to resume.

Action: Make an API call with the configuration to restart jobs that encounter this problem.

Logging Amazon Kinesis Video Streams API calls with AWS CloudTrail

Amazon Kinesis Video Streams works with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon Kinesis Video Streams. CloudTrail captures all API calls for Amazon Kinesis Video Streams as events. The calls captured include calls from the Amazon Kinesis Video Streams console and code calls to the Amazon Kinesis Video Streams API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon Kinesis Video Streams. 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 Amazon Kinesis Video Streams, 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, including how to configure and enable it, see the *AWS CloudTrail User Guide*.

Amazon Kinesis Video Streams and CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in Amazon Kinesis Video Streams, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon Kinesis Video Streams, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

Amazon Kinesis Video Streams supports logging the following actions as events in CloudTrail log files:

- CreateStream
- DeleteStream
- DescribeStream
- GetDataEndpoint
- ListStreams
- ListTagsForStream
- TagStream
- UntagStream
- UpdateDataRetention
- UpdateStream

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

- Whether the request was made with root or 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.

Example: Amazon Kinesis Video Streams 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 CreateStream action.

```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:user/Alice",
                "accountId": "123456789012",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2018-05-25T00:16:31Z",
            "eventSource": " kinesisvideo.amazonaws.com",
            "eventName": "CreateStream",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "streamName": "VideoStream",
                "dataRetentionInHours": 2,
                "mediaType": "mediaType",
                "kmsKeyId": "arn:aws:kms::us-east-1:123456789012:alias",
  "deviceName": "my-device"
        },
            "responseElements": {
  "streamARN":arn:aws:kinesisvideo:us-east-1:123456789012:stream/VideoStream/12345"
             },
            "requestID": "db6c59f8-c757-11e3-bc3b-57923b443c1c",
            "eventID": "b7acfcd0-6ca9-4ee1-a3d7-c4e8d420d99b"
        },
        {
            "eventVersion": "1.05",
```

```
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:user/Alice",
                "accountId": "123456789012",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2018-05-25:17:06Z",
            "eventSource": " kinesisvideo.amazonaws.com",
            "eventName": "DeleteStream",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "streamARN": "arn:aws:kinesisvideo:us-east-1:012345678910:stream/
VideoStream/12345",
                "currentVersion": "keqrjeqkj9"
             },
            "responseElements": null,
            "requestID": "f0944d86-c757-11e3-b4ae-25654b1d3136",
            "eventID": "0b2f1396-88af-4561-b16f-398f8eaea596"
        },
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:user/Alice",
                "accountId": "123456789012",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2014-04-19T00:15:02Z",
            "eventSource": " kinesisvideo.amazonaws.com",
            "eventName": "DescribeStream",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "streamName": "VideoStream"
             },
            "responseElements": null,
            "requestID": "a68541ca-c757-11e3-901b-cbcfe5b3677a",
```

```
            "eventID": "22a5fb8f-4e61-4bee-a8ad-3b72046b4c4d"
        },
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:user/Alice",
                "accountId": "123456789012",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2014-04-19T00:15:03Z",
            "eventSource": "kinesisvideo.amazonaws.com",
            "eventName": "GetDataEndpoint",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "streamName": "VideoStream",
                "aPIName": "LIST_FRAGMENTS"
"
            },
            "responseElements": null,
            "requestID": "a6e6e9cd-c757-11e3-901b-cbcfe5b3677a",
            "eventID": "dcd2126f-c8d2-4186-b32a-192dd48d7e33"
        },
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:user/Alice",
                "accountId": "123456789012",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2018-05-25T00:16:56Z",
            "eventSource": "kinesisvideo.amazonaws.com",
            "eventName": "ListStreams",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
```

```
                "maxResults": 100,
                "streamNameCondition": {"comparisonValue":"MyVideoStream"
 comparisonOperator":"BEGINS_WITH"}}
            },
            "responseElements": null,
            "requestID": "e9f9c8eb-c757-11e3-bf1d-6948db3cd570",
            "eventID": "77cf0d06-ce90-42da-9576-71986fec411f"
        }
    ]
}
```

Kinesis Video Streams service quotas

Kinesis Video Streams has the following service quotas:

> **Important**
>
> The following service quotas are either soft **[s]**, which can be upgraded by submitting a support ticket, or hard **[h]**, which can't be increased. You will see [s] and [h] next to individual service quota in the tables below.

Control plane API service quotas

The following section describes service quotas for control plane APIs. TPS stands for *transactions per second*.

When an account-level or resource-level request limit is reached, a `ClientLimitExceededException` is thrown.

Control plane API service quotas

API	Account limit: Request	Account limit: Streams	Stream-level limit	Relevant exceptions and notes
the section called “CreateStream”	50 TPS [s]	10000 streams per account [s] in US East (N. Virginia) and US West (Oregon) Regions. 5000 streams per account [s] in all other	N/A	Devices, CLIs, SDK-driven access, and the console can all invoke this API. Only one API call succeeds if the stream doesn’t already exist.

API	Account limit: Request	Account limit: Streams	Stream-level limit	Relevant exceptions and notes
		supported Regions. **Note** This limit can be increased up to 100,000 (or more) streams per account [s]. Sign in to the AWS Management Console at https://console.aws.amazon.com/ and submit a		

API	Account limit: Request	Account limit: Streams	Stream-level limit	Relevant exceptions and notes
		service limit increase case for Kinesis Video Streams to request an increase of this limit.		
the section called "DeleteEdgeConfiguration"	10 TPS [h]	N/A	1 TPS [h]	
the section called "DeleteStream"	50 TPS [h]	N/A	5 TPS [h]	
the section called "DescribeEdgeConfiguration"	50 TPS [h]	N/A	5 TPS [h]	

API	Account limit: Request	Account limit: Streams	Stream-level limit	Relevant exceptions and notes
the section called "DescribeImageGenerationConfiguration"	50 TPS [h]	N/A	5 TPS [h]	
the section called "DescribeMappedResourceConfiguration"	50 TPS [h]	N/A	5 TPS [h]	
the section called "DescribeNotificationConfiguration"	50 TPS [h]	N/A	5 TPS [h]	
the section called "DescribeStream"	300 TPS [h]	N/A	5 TPS [h]	
the section called "GetDataEndpoint"	300 TPS [h]	N/A	5 TPS [h]	Called every 45 minutes to refresh the streaming token for most `PutMedia`/`GetMedia` use cases. Caching data endpoints is safe if the application reloads them on failure.

API	Account limit: Request	Account limit: Streams	Stream-level limit	Relevant exceptions and notes
the section called "ListEdgeAgentConfigurations"	50 TPS [h]	N/A	N/A	
the section called "ListStreams"	50 TPS [h]	N/A	N/A	
the section called "ListTagsForStream"	50 TPS [h]	N/A	5 TPS [h]	
the section called "StartEdgeConfigurationUpdate"	10 TPS [h]	N/A	1 TPS [h]	
the section called "TagStream"	50 TPS [h]	N/A	5 TPS [h]	
the section called "UntagStream"	50 TPS [h]	N/A	5 TPS [h]	
the section called "UpdateDataRetention"	50 TPS [h]	N/A	5 TPS [h]	

API	Account limit: Request	Account limit: Streams	Stream-level limit	Relevant exceptions and notes
the section called "UpdateImageGenerationConfiguration"	50 TPS [h]	N/A	5 TPS [h]	
the section called "UpdateNotificationConfiguration"	50 TPS [h]	N/A	5 TPS [h]	
the section called "UpdateStream"	50 TPS [h]	N/A	5 TPS [h]	

Media and archived-media API service quotas

The following section describes service quotas for media and archived media APIs.

When an account-level or resource-level request limit is reached, a `ClientLimitExceededException` is thrown.

When a connection-level limit is reached, a `ConnectionLimitExceededException` is thrown.

The following errors or acks are thrown when a fragment-level limit is reached:

- A `MIN_FRAGMENT_DURATION_REACHED` ack is returned for a fragment below the minimum duration.
- A `MAX_FRAGMENT_DURATION_REACHED` ack is returned for a fragment above the maximum duration.

- A `MAX_FRAGMENT_SIZE` ack is returned for a fragment above the maximum data size.
- A `FragmentLimitExceeded` exception is thrown if a fragment limit is reached in a `GetMediaForFragmentList` operation.

Data plane API service quotas

API	Stream-level limit	Connection-level limit	Bandwidth limit	Fragment-level limit	Relevant exceptions and notes
the section called "PutMedia"	5 TPS [h]	1 [s]	12.5 MB/second, or 100 Mbps [s] **per stream**	• Minimum fragment duration: 1 second [h] • Maximum fragment duration: 20 seconds [h] • Maximum fragment size: 50 MB [h] • Maximum number of tracks: 3 [s] • Maximum fragments sent per second: 5 [h]	A typical `PutMedia` request contains data for several seconds, resulting in a lower TPS per stream. If there are multiple concurrent connections that exceed quotas, the last connection is accepted.

API	Stream-level limit	Connection-level limit	Bandwidth limit	Fragment-level limit	Relevant exceptions and notes
				• Maximum fragment metadata limit: 10 tags [h]	
[the section called "GetClip"]	N/A	N/A	100 MB size limit [h]	Maximum number of fragments: 200 [h]	
[the section called "GetDASHStreamingSessionURL"]	25 TPS [h]	N/A	N/A	N/A	
[the section called "GetHLSStreamingSessionURL"]	25 TPS [h]	N/A	N/A	N/A	

API	Stream-level limit	Connection-level limit	Bandwidth limit	Fragment-level limit	Relevant exceptions and notes
the section called “GetImages”	N/A	N/A	100 MB [h]	N/A	Maximum number of images per request is 100 [h]. **Note** The minimum value for `SamplingInterval` is 200 milliseconds (ms), which is 5 images per second.

API	Stream-level limit	Connection-level limit	Bandwidth limit	Fragment-level limit	Relevant exceptions and notes
the section called "GetMedia"	5 TPS [h]	3 [s]	25 MB/s or 200 Mbps [s]	Maximum fragments sent per second: 6 [h]	A unique consuming client shouldn't need more than two or three TPS because after the connection is established, the application should read continuously. If a typical fragment is approximately 5 MB, this limit means ~75 MBps per Kinesis video stream. Such a stream would have an outgoing bitrate of 2x the streams' maximum incoming bitrate. **Note** `GetMedia` isn't used for HLS/DASH playback.
the section called "GetMediaForFragmentList"	N/A	5 [s]	25 MB/s or 200 Mbps [s]	Maximum number of fragments: 1000 [h]	Five fragment-based consuming applications can concurrently invoke `GetMediaForFragmentList` . Further connections are rejected.

Video playback protocol API service quotas

API	Session-level limit	Fragment-level limit
GetDASHManifestPlaylist	5 TPS [h]	Maximum number of fragments per playlist: 5000 [h]
GetHLSMasterPlaylist	5 TPS [h]	N/A
GetHLSMediaPlaylist	5 TPS [h]	Maximum number of fragments per playlist: 5000 [h]
GetMP4InitFragment	5 TPS [h]	N/A
GetMP4MediaFragment	20 TPS [h]	N/A
GetTSFragment	20 TPS [h]	N/A

Fragment-metadata and fragment-media quotas

Kinesis Video Streams APIs for accessing archived media are throttled based on the number of fragments requested rather than the number of API calls. APIs are rate-limited by both the number of fragment metadata and the number of fragment media that's requested. The fragment metadata and fragment media quotas are applied per stream. In other words, requests for fragment metadata or media in one stream don't apply to the quotas of another stream. However, within a given stream, each quota is shared across multiple APIs. This means that, for a given stream, requests for fragments across different APIs consume from the same quota. When either the fragment metadata or fragment media quota for a stream is exceeded, the API returns a `ClientLimitExceededException`. The following tables show how the APIs consume from each of the two types of quota. For the second column in these tables, assume that if a stream has a quota of N, that means the APIs have N points to consume from that quota type for that stream. The `GetClip` API appears in both tables.

Fragment metadata quota consumption

API	Number of quota points consumed per request	Shared quota (N)
the section called "ListFragments"	Value of the `MaxResults` parameter	10000 quota points per second, per stream [h]
the section called "GetClip"	Number of fragments in the resulting clip	
`GetHLSMediaPlaylist`	Value of the `MaxMediaPlaylistFragmentResults` parameter	
`GetDASHManifest`	Value of the `MaxManifestFragmentResults` parameter	
the section called "GetImages"	Value of 400 + max number of images requested	

Fragment media quota consumption

API	Number of quota points consumed per request	Shared quota (N)
the section called "GetMediaForFragmentList"	Number of fragments in the Fragments parameter	500 quota points per second, per stream [h]
the section called "GetClip"	Number of fragments in the resulting clip	
`GetMP4MediaFragment`	1	
`GetTSFragment`	1	
the section called "GetImages"	Max number of images requested	

For example, with a quota of 500 fragment media per second, the following call patterns for a particular stream are supported:

- 5 requests per second to `GetClip` with 100 fragments in each clip.
- 100 requests per second to `GetClip` with 5 fragments in each clip.
- 2 requests per second to `GetClip` with 100 fragments in each clip and 3 requests per second to `GetMediaForFragmentList` in each clip.
- 400 requests per second to `GetMP4MediaFragment` and 100 requests per second to `GetTSFragment`.

These quotas have an important implication regarding the number of HLS and MPEG-DASH sessions that can be supported per stream. There's no limit to the number of HLS and DASH sessions that can be in use by media players at a given time. Therefore, it's important that the playback application doesn't allow too many sessions to be in use concurrently. The following two examples describe how to determine the number of concurrent playback sessions that can be supported:

Example 1: Live streaming

In a live streaming scenario with HLS with 1 second duration fragments, an audio and video track, and `MaxMediaPlaylistFragmentResults` set to five, a media player typically makes two calls to `GetHLSMediaPlaylist` per second. One call is for the latest video metadata and another for the corresponding audio metadata. The two calls consume five fragment metadata quota points each. It also makes two calls to `GetMP4MediaFragment` per second: one call for the latest video and another for the corresponding audio. Each call consumes a single fragment media token, so two tokens are consumed in total.

In this scenario, up to 250 concurrent playback sessions can be supported. With 250 sessions, this scenario consumes 2,500 fragment metadata quota points per second (well below the 10,000 quota) and 500 fragment media quota points per second.

Example 2: On-demand playback

In an on-demand playback scenario of a past event with MPEG-DASH, an audio and video track and `MaxManifestFragmentResults` set to 1,000, a media player typically calls `GetDASHManifest` once at the start of the session (consuming 1,000 fragment metadata quota points) and it calls `GetMP4MediaFragment` at a rate of up to 5 times per second (consuming 5 fragment media quota points) until all fragments are loaded. In this scenario, up to 10 new sessions can be started per second (right at the 10,000 fragment metadata per second quota), and up to 100 sessions can be actively loading fragment media at a rate of 5 per second (right at the 500 fragment media per second quota).

You can use `ArchivedFragmentsConsumed.Metadata` and `ArchivedFragmentsConsumed.Media` to monitor the consumption of fragment metadata and fragment media quota points, respectively. For information about monitoring, see *Monitoring*.

Fragment metadata quotas

The following service quotas apply to adding fragment metadata to fragments in a Kinesis video stream:

- You can prepend up to 10 metadata items to a fragment.
- A fragment metadata *name* can be up to 128 bytes in length.
- A fragment metadata *value* can be up to 256 bytes in length.
- A fragment metadata *name* can't begin with the string "`AWS`". If such a metadata item is added, the `putFragmentMetadata` method in the PIC returns a `STATUS_INVALID_METADATA_NAME` error (error code `0x52000077`). Your application can then either ignore the error (the PIC doesn't add the metadata item), or respond to the error.

Stream tags

These metadata key-value pairs apply to the overall Kinesis Video Streams resource, not individual fragments contained within a Kinesis video stream.

Each Kinesis video stream supports up to 50 tags.

See the section called “TagStream” for restrictions on the keys and values of stream tags.

Troubleshooting Kinesis Video Streams

Use the following information to troubleshoot common issues encountered with Amazon Kinesis Video Streams.

Topics

General issues

This section describes general issues that you might encounter when working with Kinesis Video Streams.

Issues

Latency too high

Latency might be caused by the duration of fragments that are sent to the Kinesis Video Streams service. One way to reduce the latency between the producer and the service is to configure the media pipeline to produce shorter fragment durations.

To reduce the number of frames sent in each fragment, reduce the following value in `kinesis_video_gstreamer_sample_app.cpp`:

```
g_object_set(G_OBJECT (data.encoder), "bframes", 0, "key-int-max", 45, "bitrate", 512,
 NULL);
```

> **Note**
>
> Latencies are higher in the Mozilla Firefox browser due to the internal implementation of video rendering.

API issues

This section describes API issues that you might encounter when working with Kinesis Video Streams.

Issues

Error: "Unknown options"

`GetMedia` and `GetMediaForFragmentList` can fail with the following error:

```
Unknown options: <filename>.mkv
```

This error occurs if you configured the AWS CLI with an `output` type of `json`. Reconfigure the AWS CLI with the default output type (none). For information about configuring the AWS CLI, see configure in the *AWS CLI Command Reference*.

Error: "Unable to determine service/operation name to be authorized"

`GetMedia` can fail with the following error:

```
Unable to determine service/operation name to be authorized
```

This error might occur if the endpoint is not properly specified. When you're getting the endpoint, be sure to include the following parameter in the `GetDataEndpoint` call, depending on the API to be called:

```
--api-name GET_MEDIA
--api-name PUT_MEDIA
--api-name GET_MEDIA_FOR_FRAGMENT_LIST
--api-name LIST_FRAGMENTS
```

Error: "Failed to put a frame in the stream"

`PutMedia` can fail with the following error:

```
Failed to put a frame in the stream
```

This error might occur if connectivity or permissions are not available to the service. Run the following in the AWS CLI, and verify that the stream information can be retrieved:

```
aws kinesisvideo describe-stream --stream-name StreamName --endpoint https://
ServiceEndpoint.kinesisvideo.region.amazonaws.com
```

If the call fails, see [Troubleshooting AWS CLI Errors](#) for more information.

Error: "Service closed connection before final AckEvent was received"

`PutMedia` can fail with the following error:

```
com.amazonaws.SdkClientException: Service closed connection before final AckEvent was
 received
```

This error might occur if `PushbackInputStream` is improperly implemented. Verify that the `unread()` methods are correctly implemented.

Error: "STATUS_STORE_OUT_OF_MEMORY"

`PutMedia` can fail with the following error:

```
The content store is out of memory.
```

This error occurs when the content store is not allocated with sufficient size. To increase the size of the content store, increase the value of `StorageInfo.storageSize`. For more information, see [StorageInfo](#).

HLS issues

If your video stream doesn't play back correctly, see the section called “Troubleshooting HLS issues”.

Java issues

This section describes how to troubleshoot common Java issues encountered when working with Kinesis Video Streams.

Issues

- Enabling Java logs

Enabling Java logs

To troubleshoot issues with Java samples and libraries, it's helpful to enable and examine the debug logs. To enable debug logs, do the following:

1. Add `log4j` to the `pom.xml` file, in the `dependencies` node:

   ```
   <dependency>
       <groupId>log4j</groupId>
       <artifactId>log4j</artifactId>
       <version>1.2.17</version>
   </dependency>
   ```

2. In the `target/classes` directory, create a file named `log4j.properties` with the following contents:

   ```
   # Root logger option
   log4j.rootLogger=DEBUG, stdout

   # Redirect log messages to console
   log4j.appender.stdout=org.apache.log4j.ConsoleAppender
   log4j.appender.stdout.Target=System.out
   log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
   log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:
   %L - %m%n
   ```

```
log4j.logger.org.apache.http.wire=DEBUG
```

The debug logs then print to the IDE console.

Producer Library issues

This section describes issues that you might encounter when using the Producer Libraries.

Issues

- Cannot compile the Producer SDK
- Video stream does not appear in the console
- Error: "Security token included in the request is invalid" when streaming data using the GStreamer demo application
- Error: "Failed to submit frame to Kinesis Video client"
- GStreamer application stops with "streaming stopped, reason not-negotiated" message on OS X
- Error: "Failed to allocate heap" when creating Kinesis Video Client in GStreamer demo on Raspberry Pi
- Error: "Illegal Instruction" when running GStreamer demo on Raspberry Pi
- Camera fails to load on Raspberry Pi
- Camera can't be found on macOS High Sierra
- jni.h file not found when compiling on macOS High Sierra
- Curl errors when running the GStreamer demo application
- Timestamp/range assertion at runtime on Raspberry Pi
- Assertion on gst_value_set_fraction_range_full on Raspberry Pi
- STATUS_MKV_INVALID_ANNEXB_NALU_IN_FRAME_DATA (0x3200000d) error on Android
- Maximum fragment duration was reached error
- "Invalid thing name passed" error when using IoT authorization

Cannot compile the Producer SDK

Verify that the required libraries are in your path. To verify this, use the following command:

```
env | grep LD_LIBRARY_PATH
```

```
LD_LIBRARY_PATH=/home/local/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp/
kinesis-video-native-build/downloads/local/lib
```

Video stream does not appear in the console

To display your video stream in the console, it must be encoded using H.264 in AvCC format. If your stream is not displayed, verify the following:

- Your NAL Adaptation Flags are set to `NAL_ADAPTATION_ANNEXB_NALS | NAL_ADAPTATION_ANNEXB_CPD_NALS` if the original stream is in Annex-B format. This is the default value in the `StreamDefinition` constructor.
- You are providing the codec private data correctly. For H.264, this is the sequence parameter set (SPS) and picture parameter set (PPS). Depending on your media source, this data may be retrieved from the media source separately or encoded into the frame.

  Many elementary streams are in the following format, where `Ab` is the Annex-B start code (001 or 0001):

  ```
  Ab(Sps)Ab(Pps)Ab(I-frame)Ab(P/B-frame) Ab(P/B-frame)…. Ab(Sps)Ab(Pps)Ab(I-frame)Ab(P/
  B-frame) Ab(P/B-frame)
  ```

  The CPD (Codec Private Data), if H.264 is in the stream as SPS and PPS, can be adapted to the AvCC format. Unless the media pipeline gives the CPD separately, the application can extract the CPD from the frame by looking for the first Idr frame (which should contain the SPS and PPS), extract the two NALUs (which will be Ab(Sps)Ab(Pps)) and set it in the CPD in `StreamDefinition`.

Error: "Security token included in the request is invalid" when streaming data using the GStreamer demo application

If this error occurs, there is an issue with your credentials. Verify the following:

- If you are using temporary credentials, you must specify the session token.
- Verify that your temporary credentials are not expired.
- Verify that you have the proper rights set up.
- On macOS, verify that you do not have credentials cached in Keychain.

Error: "Failed to submit frame to Kinesis Video client"

If this error occurs, the timestamps are not properly set in the source stream. Try the following:

- Use the latest SDK sample, which might have an update that fixes your issue.
- Set the high-quality stream to a higher bitrate, and fix any jitter in the source stream if the camera supports doing so.

GStreamer application stops with "streaming stopped, reason not-negotiated" message on OS X

Streaming may stop on OS X with the following message:

```
Debugging information: gstbasesrc.c(2939): void gst_base_src_loop(GstPad *) (): /
GstPipeline:test-pipeline/GstAutoVideoSrc:source/GstAVFVideoSrc:source-actual-src-
avfvide:
streaming stopped, reason not-negotiated (-4)
```

A possible workaround for this is to remove the frame rate parameters from the `gst_caps_new_simple` call in `kinesis_video_gstreamer_sample_app.cpp`:

```
GstCaps *h264_caps = gst_caps_new_simple("video/x-h264",
                                             "profile", G_TYPE_STRING, "baseline",
                                             "stream-format", G_TYPE_STRING, "avc",
                                             "alignment", G_TYPE_STRING, "au",
                                             "width", GST_TYPE_INT_RANGE, 320, 1920,
                                             "height", GST_TYPE_INT_RANGE, 240, 1080,
                                             "framerate", GST_TYPE_FRACTION_RANGE, 0,
 1, 30, 1,
                                             NULL);
```

Error: "Failed to allocate heap" when creating Kinesis Video Client in GStreamer demo on Raspberry Pi

The GStreamer sample application tries to allocate 512 MB of RAM, which might not be available on your system. You can reduce this allocation by reducing the following value in `KinesisVideoProducer.cpp`:

```
device_info.storageInfo.storageSize = 512 * 1024 * 1024;
```

Error: "Illegal Instruction" when running GStreamer demo on Raspberry Pi

If you encounter the following error when running the GStreamer demo, verify that you have compiled the application for the correct version of your device. (For example, verify that you're not compiling for Raspberry Pi 3 when you're running on Raspberry Pi 2.)

```
INFO - Initializing curl.
Illegal instruction
```

Camera fails to load on Raspberry Pi

To check whether the camera is loaded, run the following:

```
ls /dev/video*
```

If nothing is found, run the following:

```
vcgencmd get_camera
```

The output should look similar to the following:

```
supported=1 detected=1
```

If the driver does not detect the camera, do the following:

1. Check the physical camera setup and verify that it's connected properly.
2. Run the following to upgrade the firmware:

   ```
   sudo rpi-update
   ```

3. Restart the device.
4. Run the following to load the driver:

   ```
   sudo modprobe bcm2835-v4l2
   ```

5. Verify that the camera was detected:

```
ls /dev/video*
```

Camera can't be found on macOS High Sierra

On macOS High Sierra, the demo application can't find the camera if more than one camera is available.

jni.h file not found when compiling on macOS High Sierra

To resolve this error, update your installation of Xcode to the latest version.

Curl errors when running the GStreamer demo application

To resolve curl errors when you run the GStreamer demo application, copy this certificate file to /etc/ssl/cert.pem.

Timestamp/range assertion at runtime on Raspberry Pi

If a timestamp range assertion occurs at runtime, update the firmware and restart the device:

```
sudo rpi-update
$ sudo reboot
```

Assertion on gst_value_set_fraction_range_full on Raspberry Pi

The following assertion appears if the uv4l service is running:

```
gst_util_fraction_compare (numerator_start, denominator_start, numerator_end,
 denominator_end) < 0' failed
```

If this occurs, stop the uv4l service and restart the application.

STATUS_MKV_INVALID_ANNEXB_NALU_IN_FRAME_DATA (0x3200000d) error on Android

The following error appears if the NAL Adaptation Flags are incorrect for the media stream:

```
putKinesisVideoFrame(): Failed to put a frame with status code 0x3200000d
```

If this error occurs, provide the correct `.withNalAdaptationFlags` flag for your media (for example, `NAL_ADAPTATION_ANNEXB_CPD_NALS`). Provide this flag in the following line of the Android Producer Library:

https://github.com/awslabs/aws-sdk-android-samples/blob/master/AmazonKinesisVideoDemoApp/src/main/java/com/amazonaws/kinesisvideo/demoapp/fragment/StreamConfigurationFragment.java#L169

Maximum fragment duration was reached error

This error occurs when a media fragment in a stream exceeds the maximum fragment duration limit. See the maximum fragment duration limit in the the section called “Media and archived-media API service quotas” section.

To resolve this issue, try the following:

- If you are using a webcam/USB camera, do one of the following:
  - If you're using key frame-based fragmentation, then set the encoder to provide key frames within 10 seconds.
  - If you're not using key frame-based fragmentation, then when defining the stream in Step 2: Write and examine the code, set the maximum fragment duration limit to a value that's less than 10 seconds.
  - If you're using software encoders (like x264) in the GStreamer pipeline, you can set the key-int-max attribute to a value within 10 seconds. For example, set key-int-max to 60, with fps set to 30, to enable key frames every 2 seconds.
- If you're using an RPI camera, set the keyframe-interval attribute to be less than 10 seconds.
- If you're using an IP (RTSP) camera, set the GOP size to 60.

"Invalid thing name passed" error when using IoT authorization

To avoid this error (`HTTP Error 403: Response: {"message":"Invalid thing name passed"}`) when you're using IoT credentials for authorization, make sure that the value of `stream-name` (a required parameter of the `kvssink` element) is identical to the value of `iot-thingname`. For more information, see GStreamer element parameter reference.

Stream Parser Library issues

This section describes issues that you might encounter when using the Stream Parser Library.

Issues

Cannot access a single frame from the stream

To access a single frame from a streaming source in your consumer application, verify that your stream contains the correct codec private data. For information about the format of the data in a stream, see Data Model.

To learn how to use codec private data to access a frame, see the following test file on the GitHub website: KinesisVideoRendererExampleTest.java

Fragment decoding error

If your fragments are not properly encoded in an H.264 format and level that the browser supports, you might see the following error when playing your stream in the console:

```
Fragment Decoding Error
There was an error decoding the video data. Verify that the stream contains valid H.264
 content
```

If this occurs, verify the following:

- The resolution of the frames matches the resolution specified in the Codec Private Data.
- The H.264 profile and level of the encoded frames matches the profile and level specified in the Codec Private Data.
- The browser supports the profile/level combination. Most current browsers support all profile and level combinations.
- The timestamps are accurate and in the correct order, and no duplicate timestamps are being created.
- Your application is encoding the frame data using the H.264 format.

Network issues

If you see connection errors, such as "Connection Timeout" or "Connection Failed", when attempting to connect to Kinesis Video Streams, it may be due to IP address range restrictions in your networking setup.

If your setup has IP address range restrictions for Kinesis Video Streams, update your network configuration to allowlist the Kinesis Video Streams IP address ranges.

> **Important**
>
> The IP range list isn't an exhaustive list of Kinesis Video Streams IP addresses. Include the IP address ranges you see and be aware that the IP addresses may change over time.

For more information, see AWS IP ranges. To be notified when IP ranges change, follow the subscription procedure.

Document History for Amazon Kinesis Video Streams

The following table describes the important changes to the documentation since the last release of Amazon Kinesis Video Streams.

- **Latest API version:** 2017-11-29
- **Latest documentation update:** June 27, 2023

Change	Description	Date
Amazon Kinesis Video Streams Edge Agent Edge-to-Cloud Connection	New feature release. For more information, see *Edge Agent*.	June 27, 2023
Getting Started: Send Data to a Kinesis video stream	Basic tutorial for sending media data from a camera to a Kinesis video stream. For more information, see Send data to an Amazon Kinesis video stream.	January 21, 2019
Streaming metadata	You can use the Producer SDK to embed metadata in a Kinesis video stream. For more information, see Using streaming metadata with Kinesis Video Streams.	September 28, 2018
C++ Producer SDK logging	You can configure logging for C++ Producer SDK applicati ons. For more information, see Using logging with the C+ + Producer SDK.	July 18, 2018
HLS video streaming	You can now view a Kinesis video stream using HTTP Live Streaming. For more	July 13, 2018

Change	Description	Date
	information, see Kinesis Video Streams playback.	
Streaming from an RTSP source	Sample application for Kinesis Video Streams that runs in a Docker container and streams video from an RTSP source. For more information, see RTSP and Docker.	June 20, 2018
C++ Producer SDK GStreamer Plugin	Shows how to build the C ++ Producer Library to use as a GStreamer destination. For more information, see GStreamer Plugin - kvssink.	June 15, 2018
Producer SDK callbacks reference documentation	Reference documentation for the callbacks used by the Kinesis Video Streams Producer Libraries. For more information, see Producer SDK callbacks.	June 12, 2018
System requirements	Documentation for memory and storage requirements for producer devices and SDK. For more information, see Kinesis Video Streams system requirements.	May 30, 2018
CloudTrail support	Documentation for using CloudTrail to monitor API usage. For more information, see Logging Amazon Kinesis Video Streams API calls with AWS CloudTrail.	May 24, 2018

Change	Description	Date
Producer SDK structures reference documentation	Reference documentation for the structures used by the Kinesis Video Streams Producer Libraries. For more information, see Producer SDK structures and Kinesis video stream structures.	May 7, 2018
Renderer example documentation	Documentation for the Renderer example application, which shows how to decode and display frames from a Kinesis video stream. For more information, see Example: Parsing and rendering Kinesis Video Streams fragments.	March 15, 2018
Producer SDK Limits reference documentation	Information about limits for operations in the C++ Producer Library. For more information, see Producer SDK limits.	March 13, 2018
Monitoring	Information about monitoring Kinesis Video Streams metrics and API calls using Amazon CloudWatch and AWS CloudTrail. For more information, see Monitoring Amazon Kinesis Video Streams.	February 5, 2018

Change	Description	Date
Network Abstraction Layer (NAL) adaptation flag reference	Information about setting NAL adaptation flags when consuming streaming video. For more information, see NAL Adaptation Flags.	January 15, 2018
Android support for streaming video	Kinesis Video Streams now supports streaming video from Android devices. For more information, see Android Producer Library.	January 12, 2018
Kinesis Video example documentation	Documentation for the Kinesis Video example application, which shows how to use the Kinesis video stream Parser Library in an application. For more information, see KinesisVideoExample.	January 9, 2018
Kinesis Video Streams documentation released	This is the initial release of the *Amazon Kinesis Video Streams Developer Guide*.	November 29, 2017

API Reference

The sections under this node contain the API Reference documentation. Use the table of contents in the left pane to go to the different API reference sections.

Actions

The following actions are supported by Amazon Kinesis Video Streams:

- CreateSignalingChannel
- CreateStream
- DeleteEdgeConfiguration
- DeleteSignalingChannel
- DeleteStream
- DescribeEdgeConfiguration
- DescribeImageGenerationConfiguration
- DescribeMappedResourceConfiguration
- DescribeMediaStorageConfiguration
- DescribeNotificationConfiguration
- DescribeSignalingChannel
- DescribeStream
- GetDataEndpoint
- GetSignalingChannelEndpoint
- ListEdgeAgentConfigurations
- ListSignalingChannels
- ListStreams
- ListTagsForResource
- ListTagsForStream
- StartEdgeConfigurationUpdate
- TagResource
- TagStream
- UntagResource

- UntagStream
- UpdateDataRetention
- UpdateImageGenerationConfiguration
- UpdateMediaStorageConfiguration
- UpdateNotificationConfiguration
- UpdateSignalingChannel
- UpdateStream

The following actions are supported by Amazon Kinesis Video Streams Media:

- GetMedia
- PutMedia

The following actions are supported by Amazon Kinesis Video Streams Archived Media:

- GetClip
- GetDASHStreamingSessionURL
- GetHLSStreamingSessionURL
- GetImages
- GetMediaForFragmentList
- ListFragments

The following actions are supported by Amazon Kinesis Video Signaling Channels:

- GetIceServerConfig
- SendAlexaOfferToMaster

The following actions are supported by Amazon Kinesis Video WebRTC Storage:

- JoinStorageSession

Amazon Kinesis Video Streams

The following actions are supported by Amazon Kinesis Video Streams:

- CreateSignalingChannel
- CreateStream
- DeleteEdgeConfiguration
- DeleteSignalingChannel
- DeleteStream
- DescribeEdgeConfiguration
- DescribeImageGenerationConfiguration
- DescribeMappedResourceConfiguration
- DescribeMediaStorageConfiguration
- DescribeNotificationConfiguration
- DescribeSignalingChannel
- DescribeStream
- GetDataEndpoint
- GetSignalingChannelEndpoint
- ListEdgeAgentConfigurations
- ListSignalingChannels
- ListStreams
- ListTagsForResource
- ListTagsForStream
- StartEdgeConfigurationUpdate
- TagResource
- TagStream
- UntagResource
- UntagStream
- UpdateDataRetention
- UpdateImageGenerationConfiguration
- UpdateMediaStorageConfiguration
- UpdateNotificationConfiguration
- UpdateSignalingChannel
- UpdateStream

CreateSignalingChannel

Service: Amazon Kinesis Video Streams

Creates a signaling channel.

`CreateSignalingChannel` is an asynchronous operation.

Request Syntax

```
POST /createSignalingChannel HTTP/1.1
Content-type: application/json

{
   "ChannelName": "string",
   "ChannelType": "string",
   "SingleMasterConfiguration": {
      "MessageTtlSeconds": number
   },
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ChannelName

A name for the signaling channel that you are creating. It must be unique for each AWS account and AWS Region.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: Yes

ChannelType

A type of the signaling channel that you are creating. Currently, `SINGLE_MASTER` is the only supported channel type.

Type: String

Valid Values: `SINGLE_MASTER | FULL_MESH`

Required: No

SingleMasterConfiguration

A structure containing the configuration for the `SINGLE_MASTER` channel type.

Type: SingleMasterConfiguration object

Required: No

Tags

A set of tags (key-value pairs) that you want to associate with this channel.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 50 items.

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "ChannelARN": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ChannelARN

The Amazon Resource Name (ARN) of the created channel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

AccountChannelLimitExceededException

You have reached the maximum limit of active signaling channels for this AWS account in this region.

HTTP Status Code: 400

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

TagsPerResourceExceededLimitException

You have exceeded the limit of tags that you can associate with the resource. A Kinesis video stream can support up to 50 tags.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateStream

Service: Amazon Kinesis Video Streams

Creates a new Kinesis video stream.

When you create a new stream, Kinesis Video Streams assigns it a version number. When you change the stream's metadata, Kinesis Video Streams updates the version.

`CreateStream` is an asynchronous operation.

For information about how the service works, see How it Works.

You must have permissions for the `KinesisVideo:CreateStream` action.

Request Syntax

```
POST /createStream HTTP/1.1
Content-type: application/json

{
   "DataRetentionInHours": number,
   "DeviceName": "string",
   "KmsKeyId": "string",
   "MediaType": "string",
   "StreamName": "string",
   "Tags": {
      "string" : "string"
   }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

DataRetentionInHours

The number of hours that you want to retain the data in the stream. Kinesis Video Streams retains the data in a data store that is associated with the stream.

The default value is 0, indicating that the stream does not persist data.

When the `DataRetentionInHours` value is 0, consumers can still consume the fragments that remain in the service host buffer, which has a retention time limit of 5 minutes and a retention memory limit of 200 MB. Fragments are removed from the buffer when either limit is reached.

Type: Integer

Valid Range: Minimum value of 0.

Required: No

DeviceName

The name of the device that is writing to the stream.

> **Note**
> In the current implementation, Kinesis Video Streams does not use this name.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

KmsKeyId

The ID of the AWS Key Management Service (AWS KMS) key that you want Kinesis Video Streams to use to encrypt stream data.

If no key ID is specified, the default, Kinesis Video-managed key (`AWS/kinesisvideo`) is used.

For more information, see DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `.+`

Required: No

MediaType

The media type of the stream. Consumers of the stream can use this information when processing the stream. For more information about media types, see Media Types. If you choose to specify the `MediaType`, see Naming Requirements for guidelines.

Example valid values include "video/h264" and "video/h264,audio/aac".

This parameter is optional; the default value is `null` (or empty in JSON).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[\w\-\.\+]+/[\w\-\.\+]+(,[\w\-\.\+]+/[\w\-\.\+]+)*`

Required: No

StreamName

A name for the stream that you are creating.

The stream name is an identifier for the stream, and must be unique for each account and region.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: Yes

Tags

A list of tags to associate with the specified stream. Each tag is a key-value pair (the value is optional).

Type: String to string map

Map Entries: Maximum number of 50 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[\p{L}\p{Z}\p{N}_.:/=+\-@]*`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamARN": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

StreamARN

The Amazon Resource Name (ARN) of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Errors

For information about the errors that are common to all actions, see Common Errors.

AccountStreamLimitExceededException

The number of streams created for the account is too high.

HTTP Status Code: 400

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

DeviceStreamLimitExceededException

Not implemented.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

InvalidDeviceException

Not implemented.

HTTP Status Code: 400

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

TagsPerResourceExceededLimitException

You have exceeded the limit of tags that you can associate with the resource. A Kinesis video stream can support up to 50 tags.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteEdgeConfiguration

Service: Amazon Kinesis Video Streams

An asynchronous API that deletes a stream's existing edge configuration, as well as the corresponding media from the Edge Agent.

When you invoke this API, the sync status is set to `DELETING`. A deletion process starts, in which active edge jobs are stopped and all media is deleted from the edge device. The time to delete varies, depending on the total amount of stored media. If the deletion process fails, the sync status changes to `DELETE_FAILED`. You will need to re-try the deletion.

When the deletion process has completed successfully, the edge configuration is no longer accessible.

> **Note**
>
> This API isn't available in the AWS Africa (Cape Town) region, af-south-1.

Request Syntax

```
POST /deleteEdgeConfiguration HTTP/1.1
Content-type: application/json

{
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

StreamARN

The Amazon Resource Name (ARN) of the stream. Specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream from which to delete the edge configuration. Specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

StreamEdgeConfigurationNotFoundException

The Exception rendered when the Amazon Kinesis Video Stream can't find a stream's edge configuration that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteSignalingChannel

Service: Amazon Kinesis Video Streams

Deletes a specified signaling channel. `DeleteSignalingChannel` is an asynchronous operation. If you don't specify the channel's current version, the most recent version is deleted.

Request Syntax

```
POST /deleteSignalingChannel HTTP/1.1
Content-type: application/json

{
   "ChannelARN": "string",
   "CurrentVersion": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ChannelARN

The Amazon Resource Name (ARN) of the signaling channel that you want to delete.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: Yes

CurrentVersion

The current version of the signaling channel that you want to delete. You can obtain the current version by invoking the `DescribeSignalingChannel` or `ListSignalingChannels` API operations.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `[a-zA-Z0-9]+`

Required: No

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

VersionMismatchException

The stream version that you specified is not the latest version. To get the latest version, use the DescribeStream API.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteStream

Service: Amazon Kinesis Video Streams

Deletes a Kinesis video stream and the data contained in the stream.

This method marks the stream for deletion, and makes the data in the stream inaccessible immediately.

To ensure that you have the latest version of the stream before deleting it, you can specify the stream version. Kinesis Video Streams assigns a version to each stream. When you update a stream, Kinesis Video Streams assigns a new version number. To get the latest stream version, use the `DescribeStream` API.

This operation requires permission for the `KinesisVideo:DeleteStream` action.

Request Syntax

```
POST /deleteStream HTTP/1.1
Content-type: application/json

{
   "CurrentVersion": "string",
   "StreamARN": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

CurrentVersion

Optional: The version of the stream that you want to delete.

Specify the version as a safeguard to ensure that your are deleting the correct stream. To get the stream version, use the `DescribeStream` API.

If not specified, only the `CreationTime` is checked before deleting the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `[a-zA-Z0-9]+`

Required: No

StreamARN

The Amazon Resource Name (ARN) of the stream that you want to delete.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform this operation.

HTTP Status Code: 401

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

VersionMismatchException

The stream version that you specified is not the latest version. To get the latest version, use the DescribeStream API.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeEdgeConfiguration

Service: Amazon Kinesis Video Streams

Describes a stream’s edge configuration that was set using the `StartEdgeConfigurationUpdate` API and the latest status of the edge agent's recorder and uploader jobs. Use this API to get the status of the configuration to determine if the configuration is in sync with the Edge Agent. Use this API to evaluate the health of the Edge Agent.

> **Note**
>
> This API isn't available in the AWS Africa (Cape Town) region, af-south-1.

Request Syntax

```
POST /describeEdgeConfiguration HTTP/1.1
Content-type: application/json

{
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

StreamARN

The Amazon Resource Name (ARN) of the stream. Specify either the `StreamName`or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream whose edge configuration you want to update. Specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "CreationTime": number,
   "EdgeAgentStatus": {
      "LastRecorderStatus": {
         "JobStatusDetails": "string",
         "LastCollectedTime": number,
         "LastUpdatedTime": number,
         "RecorderStatus": "string"
      },
      "LastUploaderStatus": {
         "JobStatusDetails": "string",
         "LastCollectedTime": number,
         "LastUpdatedTime": number,
         "UploaderStatus": "string"
      }
   },
   "EdgeConfig": {
      "DeletionConfig": {
         "DeleteAfterUpload": boolean,
         "EdgeRetentionInHours": number,
         "LocalSizeConfig": {
            "MaxLocalMediaSizeInMB": number,
            "StrategyOnFullSize": "string"
         }
```

```
        },
        "HubDeviceArn": "string",
        "RecorderConfig": {
           "MediaSourceConfig": {
              "MediaUriSecretArn": "string",
              "MediaUriType": "string"
           },
           "ScheduleConfig": {
              "DurationInSeconds": number,
              "ScheduleExpression": "string"
           }
        },
        "UploaderConfig": {
           "ScheduleConfig": {
              "DurationInSeconds": number,
              "ScheduleExpression": "string"
           }
        }
     },
     "FailedStatusDetails": "string",
     "LastUpdatedTime": number,
     "StreamARN": "string",
     "StreamName": "string",
     "SyncStatus": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CreationTime

The timestamp at which a stream’s edge configuration was first created.

Type: Timestamp

EdgeAgentStatus

An object that contains the latest status details for an edge agent's recorder and uploader jobs. Use this information to determine the current health of an edge agent.

Type: EdgeAgentStatus object

EdgeConfig

A description of the stream's edge configuration that will be used to sync with the Edge Agent IoT Greengrass component. The Edge Agent component will run on an IoT Hub Device setup at your premise.

Type: EdgeConfig object

FailedStatusDetails

A description of the generated failure status.

Type: String

LastUpdatedTime

The timestamp at which a stream's edge configuration was last updated.

Type: Timestamp

StreamARN

The Amazon Resource Name (ARN) of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

StreamName

The name of the stream from which the edge configuration was updated.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

SyncStatus

The latest status of the edge configuration update.

Type: String

Valid Values: SYNCING | ACKNOWLEDGED | IN_SYNC | SYNC_FAILED | DELETING | DELETE_FAILED | DELETING_ACKNOWLEDGED

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

StreamEdgeConfigurationNotFoundException

The Exception rendered when the Amazon Kinesis Video Stream can't find a stream's edge configuration that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeImageGenerationConfiguration

Service: Amazon Kinesis Video Streams

Gets the `ImageGenerationConfiguration` for a given Kinesis video stream.

Request Syntax

```
POST /describeImageGenerationConfiguration HTTP/1.1
Content-type: application/json

{
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

StreamARN

The Amazon Resource Name (ARN) of the Kinesis video stream from which to retrieve the image generation configuration. You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream from which to retrieve the image generation configuration. You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "ImageGenerationConfiguration": {
      "DestinationConfig": {
         "DestinationRegion": "string",
         "Uri": "string"
      },
      "Format": "string",
      "FormatConfig": {
         "string" : "string"
      },
      "HeightPixels": number,
      "ImageSelectorType": "string",
      "SamplingInterval": number,
      "Status": "string",
      "WidthPixels": number
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ImageGenerationConfiguration

The structure that contains the information required for the Kinesis video stream (KVS) images delivery. If this structure is null, the configuration will be deleted from the stream.

Type: ImageGenerationConfiguration object

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python

- AWS SDK for Ruby V3

DescribeMappedResourceConfiguration

Service: Amazon Kinesis Video Streams

Returns the most current information about the stream. The `streamName` or `streamARN` should be provided in the input.

Request Syntax

```
POST /describeMappedResourceConfiguration HTTP/1.1
Content-type: application/json

{
   "MaxResults": number,
   "NextToken": "string",
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

MaxResults

The maximum number of results to return in the response.

Type: Integer

Valid Range: Fixed value of 1.

Required: No

NextToken

The token to provide in your next request, to get another batch of results.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

Required: No

StreamARN

The Amazon Resource Name (ARN) of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "MappedResourceConfigurationList": [
      {
         "ARN": "string",
         "Type": "string"
      }
   ],
   "NextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

MappedResourceConfigurationList

A structure that encapsulates, or contains, the media storage configuration properties.

Type: Array of MappedResourceConfigurationListItem objects

Array Members: Minimum number of 0 items. Maximum number of 1 item.

NextToken

The token that was used in the `NextToken`request to fetch the next set of results.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeMediaStorageConfiguration

Service: Amazon Kinesis Video Streams

Returns the most current information about the channel. Specify the ChannelName or ChannelARN in the input.

Request Syntax

```
POST /describeMediaStorageConfiguration HTTP/1.1
Content-type: application/json

{
   "ChannelARN": "string",
   "ChannelName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ChannelARN

The Amazon Resource Name (ARN) of the channel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

ChannelName

The name of the channel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "MediaStorageConfiguration": {
      "Status": "string",
      "StreamARN": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

MediaStorageConfiguration

A structure that encapsulates, or contains, the media storage configuration properties.

Type: MediaStorageConfiguration object

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeNotificationConfiguration

Service: Amazon Kinesis Video Streams

Gets the `NotificationConfiguration` for a given Kinesis video stream.

Request Syntax

```
POST /describeNotificationConfiguration HTTP/1.1
Content-type: application/json

{
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

StreamARN

The Amazon Resource Name (ARN) of the Kinesis video stream from where you want to retrieve the notification configuration. You must specify either the `StreamName` or the StreamARN.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream from which to retrieve the notification configuration. You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "NotificationConfiguration": {
      "DestinationConfig": {
         "Uri": "string"
      },
      "Status": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NotificationConfiguration

The structure that contains the information required for notifications. If the structure is null, the configuration will be deleted from the stream.

Type: NotificationConfiguration object

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeSignalingChannel

Service: Amazon Kinesis Video Streams

Returns the most current information about the signaling channel. You must specify either the name or the Amazon Resource Name (ARN) of the channel that you want to describe.

Request Syntax

```
POST /describeSignalingChannel HTTP/1.1
Content-type: application/json

{
   "ChannelARN": "string",
   "ChannelName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ChannelARN

The ARN of the signaling channel that you want to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

ChannelName

The name of the signaling channel that you want to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "ChannelInfo": {
      "ChannelARN": "string",
      "ChannelName": "string",
      "ChannelStatus": "string",
      "ChannelType": "string",
      "CreationTime": number,
      "SingleMasterConfiguration": {
         "MessageTtlSeconds": number
      },
      "Version": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ChannelInfo

A structure that encapsulates the specified signaling channel's metadata and properties.

Type: ChannelInfo object

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeStream

Service: Amazon Kinesis Video Streams

Returns the most current information about the specified stream. You must specify either the `StreamName` or the `StreamARN`.

Request Syntax

```
POST /describeStream HTTP/1.1
Content-type: application/json

{
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

StreamARN

The Amazon Resource Name (ARN) of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamInfo": {
      "CreationTime": number,
      "DataRetentionInHours": number,
      "DeviceName": "string",
      "KmsKeyId": "string",
      "MediaType": "string",
      "Status": "string",
      "StreamARN": "string",
      "StreamName": "string",
      "Version": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

StreamInfo

An object that describes the stream.

Type: StreamInfo object

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform this operation.

HTTP Status Code: 401

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetDataEndpoint

Service: Amazon Kinesis Video Streams

Gets an endpoint for a specified stream for either reading or writing. Use this endpoint in your application to read from the specified stream (using the `GetMedia` or `GetMediaForFragmentList` operations) or write to it (using the `PutMedia` operation).

> **Note**
>
> The returned endpoint does not have the API name appended. The client needs to add the API name to the returned endpoint.

In the request, specify the stream either by `StreamName` or `StreamARN`.

Request Syntax

```
POST /getDataEndpoint HTTP/1.1
Content-type: application/json

{
   "APIName": "string",
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

APIName

The name of the API action for which to get an endpoint.

Type: String

Valid Values: PUT_MEDIA | GET_MEDIA | LIST_FRAGMENTS | GET_MEDIA_FOR_FRAGMENT_LIST | GET_HLS_STREAMING_SESSION_URL | GET_DASH_STREAMING_SESSION_URL | GET_CLIP | GET_IMAGES

Required: Yes

StreamARN

The Amazon Resource Name (ARN) of the stream that you want to get the endpoint for. You must specify either this parameter or a `StreamName` in the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream that you want to get the endpoint for. You must specify either this parameter or a `StreamARN` in the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "DataEndpoint": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

DataEndpoint

The endpoint value. To read data from the stream or to write data to it, specify this endpoint in your application.

Type: String

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform this operation.

HTTP Status Code: 401

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetSignalingChannelEndpoint

Service: Amazon Kinesis Video Streams

Provides an endpoint for the specified signaling channel to send and receive messages. This API uses the `SingleMasterChannelEndpointConfiguration` input parameter, which consists of the `Protocols` and `Role` properties.

`Protocols` is used to determine the communication mechanism. For example, if you specify `WSS` as the protocol, this API produces a secure websocket endpoint. If you specify `HTTPS` as the protocol, this API generates an HTTPS endpoint. If you specify `WEBRTC` as the protocol, but the signaling channel isn't configured for ingestion, you will receive the error `InvalidArgumentException`.

`Role` determines the messaging permissions. A `MASTER` role results in this API generating an endpoint that a client can use to communicate with any of the viewers on the channel. A `VIEWER` role results in this API generating an endpoint that a client can use to communicate only with a `MASTER`.

Request Syntax

```
POST /getSignalingChannelEndpoint HTTP/1.1
Content-type: application/json

{
   "ChannelARN": "string",
   "SingleMasterChannelEndpointConfiguration": {
      "Protocols": [ "string" ],
      "Role": "string"
   }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ChannelARN

The Amazon Resource Name (ARN) of the signalling channel for which you want to get an endpoint.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: Yes

SingleMasterChannelEndpointConfiguration

A structure containing the endpoint configuration for the `SINGLE_MASTER` channel type.

Type: SingleMasterChannelEndpointConfiguration object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "ResourceEndpointList": [
      {
         "Protocol": "string",
         "ResourceEndpoint": "string"
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ResourceEndpointList

A list of endpoints for the specified signaling channel.

Type: Array of ResourceEndpointListItem objects

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListEdgeAgentConfigurations

Service: Amazon Kinesis Video Streams

Returns an array of edge configurations associated with the specified Edge Agent.

In the request, you must specify the Edge Agent `HubDeviceArn`.

> **Note**
>
> This API isn't available in the AWS Africa (Cape Town) region, af-south-1.

Request Syntax

```
POST /listEdgeAgentConfigurations HTTP/1.1
Content-type: application/json

{
   "HubDeviceArn": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

HubDeviceArn

The "Internet of Things (IoT) Thing" Arn of the edge agent.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:iot:[a-z0-9-]+:[0-9]+:thing/[a-zA-Z0-9_.-]+`

Required: Yes

MaxResults

The maximum number of edge configurations to return in the response. The default is 5.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 10.

Required: No

NextToken

If you specify this parameter, when the result of a `ListEdgeAgentConfigurations` operation is truncated, the call returns the `NextToken` in the response. To get another batch of edge configurations, provide this token in your next request.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "EdgeConfigs": [
      {
         "CreationTime": number,
         "EdgeConfig": {
            "DeletionConfig": {
               "DeleteAfterUpload": boolean,
               "EdgeRetentionInHours": number,
               "LocalSizeConfig": {
                  "MaxLocalMediaSizeInMB": number,
                  "StrategyOnFullSize": "string"
               }
            },
            "HubDeviceArn": "string",
            "RecorderConfig": {
```

```
                "MediaSourceConfig": {
                    "MediaUriSecretArn": "string",
                    "MediaUriType": "string"
                },
                "ScheduleConfig": {
                    "DurationInSeconds": number,
                    "ScheduleExpression": "string"
                }
            },
            "UploaderConfig": {
                "ScheduleConfig": {
                    "DurationInSeconds": number,
                    "ScheduleExpression": "string"
                }
            }
        },
        "FailedStatusDetails": "string",
        "LastUpdatedTime": number,
        "StreamARN": "string",
        "StreamName": "string",
        "SyncStatus": "string"
      }
   ],
   "NextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

EdgeConfigs

A description of a single stream's edge configuration.

Type: Array of ListEdgeAgentConfigurationsEdgeConfig objects

NextToken

If the response is truncated, the call returns this element with a given token. To get the next batch of edge configurations, use this token in your next request.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform this operation.

HTTP Status Code: 401

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python

- [AWS SDK for Ruby V3]

ListSignalingChannels

Service: Amazon Kinesis Video Streams

Returns an array of `ChannelInfo` objects. Each object describes a signaling channel. To retrieve only those channels that satisfy a specific condition, you can specify a `ChannelNameCondition`.

Request Syntax

```
POST /listSignalingChannels HTTP/1.1
Content-type: application/json

{
   "ChannelNameCondition": {
      "ComparisonOperator": "string",
      "ComparisonValue": "string"
   },
   "MaxResults": number,
   "NextToken": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ChannelNameCondition

Optional: Returns only the channels that satisfy a specific condition.

Type: ChannelNameCondition object

Required: No

MaxResults

The maximum number of channels to return in the response. The default is 500.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 10000.

Required: No

NextToken

If you specify this parameter, when the result of a `ListSignalingChannels` operation is truncated, the call returns the `NextToken` in the response. To get another batch of channels, provide this token in your next request.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "ChannelInfoList": [
      {
         "ChannelARN": "string",
         "ChannelName": "string",
         "ChannelStatus": "string",
         "ChannelType": "string",
         "CreationTime": number,
         "SingleMasterConfiguration": {
            "MessageTtlSeconds": number
         },
         "Version": "string"
      }
   ],
   "NextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ChannelInfoList

An array of `ChannelInfo` objects.

Type: Array of ChannelInfo objects

NextToken

If the response is truncated, the call returns this element with a token. To get the next batch of streams, use this token in your next request.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListStreams

Service: Amazon Kinesis Video Streams

Returns an array of `StreamInfo` objects. Each object describes a stream. To retrieve only streams that satisfy a specific condition, you can specify a `StreamNameCondition`.

Request Syntax

```
POST /listStreams HTTP/1.1
Content-type: application/json

{
   "MaxResults": number,
   "NextToken": "string",
   "StreamNameCondition": {
      "ComparisonOperator": "string",
      "ComparisonValue": "string"
   }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

MaxResults

The maximum number of streams to return in the response. The default is 10,000.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 10000.

Required: No

NextToken

If you specify this parameter, when the result of a `ListStreams` operation is truncated, the call returns the `NextToken` in the response. To get another batch of streams, provide this token in your next request.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

Required: No

StreamNameCondition

Optional: Returns only streams that satisfy a specific condition. Currently, you can specify only the prefix of a stream name as a condition.

Type: StreamNameCondition object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "StreamInfoList": [
      {
         "CreationTime": number,
         "DataRetentionInHours": number,
         "DeviceName": "string",
         "KmsKeyId": "string",
         "MediaType": "string",
         "Status": "string",
         "StreamARN": "string",
         "StreamName": "string",
         "Version": "string"
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken

If the response is truncated, the call returns this element with a token. To get the next batch of streams, use this token in your next request.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

StreamInfoList

An array of `StreamInfo` objects.

Type: Array of StreamInfo objects

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListTagsForResource

Service: Amazon Kinesis Video Streams

Returns a list of tags associated with the specified signaling channel.

Request Syntax

```
POST /ListTagsForResource HTTP/1.1
Content-type: application/json

{
   "NextToken": "string",
   "ResourceARN": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

NextToken

If you specify this parameter and the result of a `ListTagsForResource` call is truncated, the response includes a token that you can use in the next request to fetch the next batch of tags.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

Required: No

ResourceARN

The Amazon Resource Name (ARN) of the signaling channel for which you want to list tags.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "Tags": {
      "string" : "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken

If you specify this parameter and the result of a `ListTagsForResource` call is truncated, the response includes a token that you can use in the next request to fetch the next set of tags.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

Tags

A map of tag keys and values associated with the specified signaling channel.

Type: String to string map

Map Entries: Maximum number of 50 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[\p{L}\p{Z}\p{N}_.:/=+\-@]*`

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++

- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListTagsForStream

Service: Amazon Kinesis Video Streams

Returns a list of tags associated with the specified stream.

In the request, you must specify either the `StreamName` or the `StreamARN`.

Request Syntax

```
POST /listTagsForStream HTTP/1.1
Content-type: application/json

{
   "NextToken": "string",
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

NextToken

If you specify this parameter and the result of a `ListTagsForStream` call is truncated, the response includes a token that you can use in the next request to fetch the next batch of tags.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

Required: No

StreamARN

The Amazon Resource Name (ARN) of the stream that you want to list tags for.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream that you want to list tags for.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "Tags": {
      "string" : "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken

If you specify this parameter and the result of a `ListTags` call is truncated, the response includes a token that you can use in the next request to fetch the next set of tags.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 512.

Pattern: `[a-zA-Z0-9+/=]*`

Tags

A map of tag keys and values associated with the specified stream.

Type: String to string map

Map Entries: Maximum number of 50 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[\p{L}\p{Z}\p{N}_.:/=+\-@]*`

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

InvalidResourceFormatException

The format of the `StreamARN` is invalid.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform this operation.

HTTP Status Code: 401

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

StartEdgeConfigurationUpdate

Service: Amazon Kinesis Video Streams

An asynchronous API that updates a stream's existing edge configuration. The Kinesis Video Stream will sync the stream's edge configuration with the Edge Agent IoT Greengrass component that runs on an IoT Hub Device, setup at your premise. The time to sync can vary and depends on the connectivity of the Hub Device. The `SyncStatus` will be updated as the edge configuration is acknowledged, and synced with the Edge Agent.

If this API is invoked for the first time, a new edge configuration will be created for the stream, and the sync status will be set to `SYNCING`. You will have to wait for the sync status to reach a terminal state such as: `IN_SYNC`, or `SYNC_FAILED`, before using this API again. If you invoke this API during the syncing process, a `ResourceInUseException` will be thrown. The connectivity of the stream's edge configuration and the Edge Agent will be retried for 15 minutes. After 15 minutes, the status will transition into the `SYNC_FAILED` state.

To move an edge configuration from one device to another, use [DeleteEdgeConfiguration]() to delete the current edge configuration. You can then invoke StartEdgeConfigurationUpdate with an updated Hub Device ARN.

> **Note**
>
> This API isn't available in the AWS Africa (Cape Town) region, af-south-1.

Request Syntax

```
POST /startEdgeConfigurationUpdate HTTP/1.1
Content-type: application/json

{
   "EdgeConfig": {
      "DeletionConfig": {
         "DeleteAfterUpload": boolean,
         "EdgeRetentionInHours": number,
         "LocalSizeConfig": {
            "MaxLocalMediaSizeInMB": number,
            "StrategyOnFullSize": "string"
         }
      },
```

```
         "HubDeviceArn": "string",
         "RecorderConfig": {
            "MediaSourceConfig": {
               "MediaUriSecretArn": "string",
               "MediaUriType": "string"
            },
            "ScheduleConfig": {
               "DurationInSeconds": number,
               "ScheduleExpression": "string"
            }
         },
         "UploaderConfig": {
            "ScheduleConfig": {
               "DurationInSeconds": number,
               "ScheduleExpression": "string"
            }
         }
      },
      "StreamARN": "string",
      "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

EdgeConfig

The edge configuration details required to invoke the update process.

Type: EdgeConfig object

Required: Yes

StreamARN

The Amazon Resource Name (ARN) of the stream. Specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream whose edge configuration you want to update. Specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "CreationTime": number,
   "EdgeConfig": {
      "DeletionConfig": {
         "DeleteAfterUpload": boolean,
         "EdgeRetentionInHours": number,
         "LocalSizeConfig": {
            "MaxLocalMediaSizeInMB": number,
            "StrategyOnFullSize": "string"
         }
      },
      "HubDeviceArn": "string",
      "RecorderConfig": {
         "MediaSourceConfig": {
            "MediaUriSecretArn": "string",
            "MediaUriType": "string"
         },
         "ScheduleConfig": {
            "DurationInSeconds": number,
```

```
            "ScheduleExpression": "string"
         }
      },
      "UploaderConfig": {
         "ScheduleConfig": {
            "DurationInSeconds": number,
            "ScheduleExpression": "string"
         }
      }
   },
   "FailedStatusDetails": "string",
   "LastUpdatedTime": number,
   "StreamARN": "string",
   "StreamName": "string",
   "SyncStatus": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CreationTime

The timestamp at which a stream's edge configuration was first created.

Type: Timestamp

EdgeConfig

A description of the stream's edge configuration that will be used to sync with the Edge Agent IoT Greengrass component. The Edge Agent component will run on an IoT Hub Device setup at your premise.

Type: EdgeConfig object

FailedStatusDetails

A description of the generated failure status.

Type: String

LastUpdatedTime

The timestamp at which a stream's edge configuration was last updated.

Type: Timestamp

StreamARN

The Amazon Resource Name (ARN) of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

StreamName

The name of the stream from which the edge configuration was updated.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

SyncStatus

The current sync status of the stream's edge configuration. When you invoke this API, the sync status will be set to the `SYNCING` state. Use the `DescribeEdgeConfiguration` API to get the latest status of the edge configuration.

Type: String

Valid Values: `SYNCING | ACKNOWLEDGED | IN_SYNC | SYNC_FAILED | DELETING | DELETE_FAILED | DELETING_ACKNOWLEDGED`

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

NoDataRetentionException

The Stream data retention in hours is equal to zero.

HTTP Status Code: 400

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

TagResource

Service: Amazon Kinesis Video Streams

Adds one or more tags to a signaling channel. A *tag* is a key-value pair (the value is optional) that you can define and assign to AWS resources. If you specify a tag that already exists, the tag value is replaced with the value that you specify in the request. For more information, see Using Cost Allocation Tags in the *AWS Billing and Cost Management and Cost Management User Guide*.

Request Syntax

```
POST /TagResource HTTP/1.1
Content-type: application/json

{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ResourceARN

The Amazon Resource Name (ARN) of the signaling channel to which you want to add tags.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: Yes

Tags

A list of tags to associate with the specified signaling channel. Each tag is a key-value pair.

Type: Array of Tag objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

TagsPerResourceExceededLimitException

You have exceeded the limit of tags that you can associate with the resource. A Kinesis video stream can support up to 50 tags.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

TagStream

Service: Amazon Kinesis Video Streams

Adds one or more tags to a stream. A *tag* is a key-value pair (the value is optional) that you can define and assign to AWS resources. If you specify a tag that already exists, the tag value is replaced with the value that you specify in the request. For more information, see Using Cost Allocation Tags in the *AWS Billing and Cost Management and Cost Management User Guide*.

You must provide either the `StreamName` or the `StreamARN`.

This operation requires permission for the `KinesisVideo:TagStream` action.

A Kinesis video stream can support up to 50 tags.

Request Syntax

```
POST /tagStream HTTP/1.1
Content-type: application/json

{
   "StreamARN": "string",
   "StreamName": "string",
   "Tags": {
      "string" : "string"
   }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

StreamARN

The Amazon Resource Name (ARN) of the resource that you want to add the tag or tags to.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream that you want to add the tag or tags to.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Tags

A list of tags to associate with the specified stream. Each tag is a key-value pair (the value is optional).

Type: String to string map

Map Entries: Maximum number of 50 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[\p{L}\p{Z}\p{N}_.:/=+\-@]*`

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

InvalidResourceFormatException

The format of the `StreamARN` is invalid.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform this operation.

HTTP Status Code: 401

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

TagsPerResourceExceededLimitException

You have exceeded the limit of tags that you can associate with the resource. A Kinesis video stream can support up to 50 tags.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UntagResource

Service: Amazon Kinesis Video Streams

Removes one or more tags from a signaling channel. In the request, specify only a tag key or keys; don't specify the value. If you specify a tag key that does not exist, it's ignored.

Request Syntax

```
POST /UntagResource HTTP/1.1
Content-type: application/json

{
   "ResourceARN": "string",
   "TagKeyList": [ "string" ]
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ResourceARN

The Amazon Resource Name (ARN) of the signaling channel from which you want to remove tags.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: Yes

TagKeyList

A list of the keys of the tags that you want to remove.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UntagStream

Service: Amazon Kinesis Video Streams

Removes one or more tags from a stream. In the request, specify only a tag key or keys; don't specify the value. If you specify a tag key that does not exist, it's ignored.

In the request, you must provide the `StreamName` or `StreamARN`.

Request Syntax

```
POST /untagStream HTTP/1.1
Content-type: application/json

{
   "StreamARN": "string",
   "StreamName": "string",
   "TagKeyList": [ "string" ]
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

StreamARN

The Amazon Resource Name (ARN) of the stream that you want to remove tags from.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream that you want to remove tags from.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

TagKeyList

A list of the keys of the tags that you want to remove.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

InvalidResourceFormatException

The format of the `StreamARN` is invalid.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform this operation.

HTTP Status Code: 401

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateDataRetention

Service: Amazon Kinesis Video Streams

Increases or decreases the stream's data retention period by the value that you specify. To indicate whether you want to increase or decrease the data retention period, specify the `Operation` parameter in the request body. In the request, you must specify either the `StreamName` or the `StreamARN`.

This operation requires permission for the `KinesisVideo:UpdateDataRetention` action.

Changing the data retention period affects the data in the stream as follows:

- If the data retention period is increased, existing data is retained for the new retention period. For example, if the data retention period is increased from one hour to seven hours, all existing data is retained for seven hours.
- If the data retention period is decreased, existing data is retained for the new retention period. For example, if the data retention period is decreased from seven hours to one hour, all existing data is retained for one hour, and any data older than one hour is deleted immediately.

Request Syntax

```
POST /updateDataRetention HTTP/1.1
Content-type: application/json

{
   "CurrentVersion": "string",
   "DataRetentionChangeInHours": number,
   "Operation": "string",
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

CurrentVersion

The version of the stream whose retention period you want to change. To get the version, call either the `DescribeStream` or the `ListStreams` API.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `[a-zA-Z0-9]+`

Required: Yes

DataRetentionChangeInHours

The number of hours to adjust the current retention by. The value you specify is added to or subtracted from the current value, depending on the `operation`.

The minimum value for data retention is 0 and the maximum value is 87600 (ten years).

Type: Integer

Valid Range: Minimum value of 1.

Required: Yes

Operation

Indicates whether you want to increase or decrease the retention period.

Type: String

Valid Values: `INCREASE_DATA_RETENTION | DECREASE_DATA_RETENTION`

Required: Yes

StreamARN

The Amazon Resource Name (ARN) of the stream whose retention period you want to change.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream whose retention period you want to change.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform this operation.

HTTP Status Code: 401

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

VersionMismatchException

The stream version that you specified is not the latest version. To get the latest version, use the DescribeStream API.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3

- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateImageGenerationConfiguration

Service: Amazon Kinesis Video Streams

Updates the `StreamInfo` and `ImageProcessingConfiguration` fields.

Request Syntax

```
POST /updateImageGenerationConfiguration HTTP/1.1
Content-type: application/json

{
   "ImageGenerationConfiguration": {
      "DestinationConfig": {
         "DestinationRegion": "string",
         "Uri": "string"
      },
      "Format": "string",
      "FormatConfig": {
         "string" : "string"
      },
      "HeightPixels": number,
      "ImageSelectorType": "string",
      "SamplingInterval": number,
      "Status": "string",
      "WidthPixels": number
   },
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ImageGenerationConfiguration

The structure that contains the information required for the KVS images delivery. If the structure is null, the configuration will be deleted from the stream.

Type: ImageGenerationConfiguration object

Required: No

StreamARN

The Amazon Resource Name (ARN) of the Kinesis video stream from where you want to update the image generation configuration. You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream from which to update the image generation configuration. You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

NoDataRetentionException

The Stream data retention in hours is equal to zero.

HTTP Status Code: 400

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateMediaStorageConfiguration

Service: Amazon Kinesis Video Streams

Associates a `SignalingChannel` to a stream to store the media. There are two signaling modes that you can specify :

- If `StorageStatus` is enabled, the data will be stored in the `StreamARN` provided. In order for WebRTC Ingestion to work, the stream must have data retention enabled.
- If `StorageStatus` is disabled, no data will be stored, and the `StreamARN` parameter will not be needed.

> **Important**
>
> If `StorageStatus` is enabled, direct peer-to-peer (master-viewer) connections no longer occur. Peers connect directly to the storage session. You must call the `JoinStorageSession` API to trigger an SDP offer send and establish a connection between a peer and the storage session.

Request Syntax

```
POST /updateMediaStorageConfiguration HTTP/1.1
Content-type: application/json

{
   "ChannelARN": "string",
   "MediaStorageConfiguration": {
      "Status": "string",
      "StreamARN": "string"
   }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ChannelARN

The Amazon Resource Name (ARN) of the channel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: Yes

MediaStorageConfiguration

A structure that encapsulates, or contains, the media storage configuration properties.

Type: MediaStorageConfiguration object

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

NoDataRetentionException

The Stream data retention in hours is equal to zero.

HTTP Status Code: 400

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface]
- [AWS SDK for .NET]
- [AWS SDK for C++]

- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateNotificationConfiguration

Service: Amazon Kinesis Video Streams

Updates the notification information for a stream.

Request Syntax

```
POST /updateNotificationConfiguration HTTP/1.1
Content-type: application/json

{
   "NotificationConfiguration": {
      "DestinationConfig": {
         "Uri": "string"
      },
      "Status": "string"
   },
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

NotificationConfiguration

The structure containing the information required for notifications. If the structure is null, the configuration will be deleted from the stream.

Type: NotificationConfiguration object

Required: No

StreamARN

The Amazon Resource Name (ARN) of the Kinesis video stream from where you want to update the notification configuration. You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream from which to update the notification configuration. You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

NoDataRetentionException

The Stream data retention in hours is equal to zero.

HTTP Status Code: 400

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2

- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateSignalingChannel

Service: Amazon Kinesis Video Streams

Updates the existing signaling channel. This is an asynchronous operation and takes time to complete.

If the `MessageTtlSeconds` value is updated (either increased or reduced), it only applies to new messages sent via this channel after it's been updated. Existing messages are still expired as per the previous `MessageTtlSeconds` value.

Request Syntax

```
POST /updateSignalingChannel HTTP/1.1
Content-type: application/json

{
   "ChannelARN": "string",
   "CurrentVersion": "string",
   "SingleMasterConfiguration": {
      "MessageTtlSeconds": number
   }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ChannelARN

The Amazon Resource Name (ARN) of the signaling channel that you want to update.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: Yes

CurrentVersion

The current version of the signaling channel that you want to update.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `[a-zA-Z0-9]+`

Required: Yes

SingleMasterConfiguration

The structure containing the configuration for the `SINGLE_MASTER` type of the signaling channel that you want to update.

Type: SingleMasterConfiguration object

Required: No

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 401

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

VersionMismatchException

The stream version that you specified is not the latest version. To get the latest version, use the DescribeStream API.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateStream

Service: Amazon Kinesis Video Streams

Updates stream metadata, such as the device name and media type.

You must provide the stream name or the Amazon Resource Name (ARN) of the stream.

To make sure that you have the latest version of the stream before updating it, you can specify the stream version. Kinesis Video Streams assigns a version to each stream. When you update a stream, Kinesis Video Streams assigns a new version number. To get the latest stream version, use the `DescribeStream` API.

`UpdateStream` is an asynchronous operation, and takes time to complete.

Request Syntax

```
POST /updateStream HTTP/1.1
Content-type: application/json

{
   "CurrentVersion": "string",
   "DeviceName": "string",
   "MediaType": "string",
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

CurrentVersion

The version of the stream whose metadata you want to update.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: [a-zA-Z0-9]+

Required: Yes

DeviceName

The name of the device that is writing to the stream.

> **Note**
> In the current implementation, Kinesis Video Streams does not use this name.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: [a-zA-Z0-9_.-]+

Required: No

MediaType

The stream's media type. Use `MediaType` to specify the type of content that the stream contains to the consumers of the stream. For more information about media types, see Media Types. If you choose to specify the `MediaType`, see Naming Requirements.

To play video on the console, you must specify the correct video type. For example, if the video in the stream is H.264, specify `video/h264` as the `MediaType`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[\w\-\.\+]+/[\w\-\.\+]+(,[\w\-\.\+]+/[\w\-\.\+]+)*`

Required: No

StreamARN

The ARN of the stream whose metadata you want to update.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream whose metadata you want to update.

The stream name is an identifier for the stream, and must be unique for each account and region.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform this operation.

HTTP Status Code: 401

ResourceInUseException

When the input `StreamARN` or `ChannelARN` in `CLOUD_STORAGE_MODE` is already mapped to a different Kinesis Video Stream resource, or if the provided input `StreamARN` or `ChannelARN` is not in Active status, try one of the following :

1. The `DescribeMediaStorageConfiguration` API to determine what the stream given channel is mapped to.
2. The `DescribeMappedResourceConfiguration` API to determine the channel that the given stream is mapped to.
3. The `DescribeStream` or `DescribeSignalingChannel` API to determine the status of the resource.

HTTP Status Code: 400

ResourceNotFoundException

Amazon Kinesis Video Streams can't find the stream that you specified.

HTTP Status Code: 404

VersionMismatchException

The stream version that you specified is not the latest version. To get the latest version, use the DescribeStream API.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++

- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Amazon Kinesis Video Streams Media

The following actions are supported by Amazon Kinesis Video Streams Media:

- GetMedia
- PutMedia

GetMedia

Service: Amazon Kinesis Video Streams Media

Use this API to retrieve media content from a Kinesis video stream. In the request, you identify the stream name or stream Amazon Resource Name (ARN), and the starting chunk. Kinesis Video Streams then returns a stream of chunks in order by fragment number.

> **Note**
>
> You must first call the `GetDataEndpoint` API to get an endpoint. Then send the `GetMedia` requests to this endpoint using the [--endpoint-url parameter](https://docs.aws.amazon.com/cli/latest/reference/).

When you put media data (fragments) on a stream, Kinesis Video Streams stores each incoming fragment and related metadata in what is called a "chunk." For more information, see [PutMedia](https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/API_dataplane_PutMedia.html). The `GetMedia` API returns a stream of these chunks starting from the chunk that you specify in the request.

The following limits apply when using the `GetMedia` API:

- A client can call `GetMedia` up to five times per second per stream.
- Kinesis Video Streams sends media data at a rate of up to 25 megabytes per second (or 200 megabits per second) during a `GetMedia` session.

> **Note**
>
> The `GetMedia` HTTP response status code will be returned immediately, but the reading of the HTTP response payload will timeout after 3 seconds if there are no ingested fragments available for playback.

> **Note**
>
> If an error is thrown after invoking a Kinesis Video Streams media API, in addition to the HTTP status code and the response body, it includes the following pieces of information:
>
> - `x-amz-ErrorType` HTTP header – contains a more specific error type in addition to what the HTTP status code provides.

- `x-amz-RequestId` HTTP header – if you want to report an issue to AWS, the support team can better diagnose the problem if given the Request Id.

Both the HTTP status code and the ErrorType header can be utilized to make programmatic decisions about whether errors are retry-able and under what conditions, as well as provide information on what actions the client programmer might need to take in order to successfully try again.
For more information, see the **Errors** section at the bottom of this topic, as well as Common Errors.

Request Syntax

```
POST /getMedia HTTP/1.1
Content-type: application/json

{
   "StartSelector": {
      "AfterFragmentNumber": "string",
      "ContinuationToken": "string",
      "StartSelectorType": "string",
      "StartTimestamp": number
   },
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

StartSelector

Identifies the starting chunk to get from the specified stream.

Type: StartSelector object

Required: Yes

StreamARN

The ARN of the stream from where you want to get the media content. If you don't specify the `streamARN`, you must specify the `streamName`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The Kinesis video stream name from where you want to get the media content. If you don't specify the `streamName`, you must specify the `streamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-Type: ContentType

Payload
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The response returns the following HTTP headers.

ContentType

The content type of the requested media.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^[a-zA-Z0-9_\.\-]+$`

The response returns the following as the HTTP body.

Payload

The payload Kinesis Video Streams returns is a sequence of chunks from the specified stream. For more information about the chunks, see PutMedia. The chunks that Kinesis Video Streams returns in the `GetMedia` call also include the following additional Matroska (MKV) tags:

- AWS_KINESISVIDEO_CONTINUATION_TOKEN (UTF-8 string) - In the event your `GetMedia` call terminates, you can use this continuation token in your next request to get the next chunk where the last request terminated.
- AWS_KINESISVIDEO_MILLIS_BEHIND_NOW (UTF-8 string) - Client applications can use this tag value to determine how far behind the chunk returned in the response is from the latest chunk on the stream.
- AWS_KINESISVIDEO_FRAGMENT_NUMBER - Fragment number returned in the chunk.
- AWS_KINESISVIDEO_SERVER_TIMESTAMP - Server timestamp of the fragment.
- AWS_KINESISVIDEO_PRODUCER_TIMESTAMP - Producer timestamp of the fragment.

The following tags will be present if an error occurs:

- AWS_KINESISVIDEO_ERROR_CODE - String description of an error that caused GetMedia to stop.
- AWS_KINESISVIDEO_ERROR_ID: Integer code of the error.

The error codes are as follows:

- 3002 - Error writing to the stream
- 4000 - Requested fragment is not found
- 4500 - Access denied for the stream's KMS key
- 4501 - Stream's KMS key is disabled
- 4502 - Validation error on the stream's KMS key

- 4503 - KMS key specified in the stream is unavailable
- 4504 - Invalid usage of the KMS key specified in the stream
- 4505 - Invalid state of the KMS key specified in the stream
- 4506 - Unable to find the KMS key specified in the stream
- 5000 - Internal error

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

ConnectionLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client connections.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

InvalidEndpointException

Caller used wrong endpoint to write data to a stream. On receiving such an exception, the user must call `GetDataEndpoint` with `APIName` set to `PUT_MEDIA` and use the endpoint from response to invoke the next `PutMedia` call.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform an operation on the given stream, or the token has expired.

HTTP Status Code: 401

ResourceNotFoundException

Status Code: 404, The stream with the given name does not exist.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutMedia

Service: Amazon Kinesis Video Streams Media

Use this API to send media data to a Kinesis video stream.

> **Note**
>
> You must first call the `GetDataEndpoint` API to get an endpoint. Then send the `PutMedia` requests to this endpoint using the --endpoint-url parameter.

In the request, you use the HTTP headers to provide parameter information, for example, stream name, timestamp, and whether the timestamp value is absolute or relative to when the producer started recording. You use the request body to send the media data. Kinesis Video Streams supports only the Matroska (MKV) container format for sending media data using this API.

You have the following options for sending data using this API:

- Send media data in real time: For example, a security camera can send frames in real time as it generates them. This approach minimizes the latency between the video recording and data sent on the wire. This is referred to as a continuous producer. In this case, a consumer application can read the stream in real time or when needed.
- Send media data offline (in batches): For example, a body camera might record video for hours and store it on the device. Later, when you connect the camera to the docking port, the camera can start a `PutMedia` session to send data to a Kinesis video stream. In this scenario, latency is not an issue.

When using this API, note the following considerations:

- You must specify either `streamName` or `streamARN`, but not both.
- To be able to play the media on the console or via HLS, track 1 of each fragment should contain h.264 encoded video, the CodecID in the fragment metadata should be "V_MPEG/ISO/AVC", and the fragment metadata should include AVCC formatted h.264 codec private data. Optionally, track 2 of each fragment should contain AAC encoded audio, the CodecID in the fragment metadata should be "A_AAC", and the fragment metadata should include AAC codec private data.
- You might find it easier to use a single long-running `PutMedia` session and send a large number of media data fragments in the payload. For each fragment received, Kinesis Video Streams

sends one or more acknowledgements. Potential network considerations might cause you to not get all these acknowledgements as they are generated.

- You might choose multiple consecutive `PutMedia` sessions, each with fewer fragments to ensure that you get all acknowledgements from the service in real time.

> **Note**
>
> If you send data to the same stream on multiple simultaneous `PutMedia` sessions, the media fragments get interleaved on the stream. You should make sure that this is OK in your application scenario.

The following limits apply when using the `PutMedia` API:

- A client can call `PutMedia` up to five times per second per stream.
- A client can send up to five fragments per second per stream.
- Kinesis Video Streams reads media data at a rate of up to 12.5 MB/second, or 100 Mbps during a `PutMedia` session.

Note the following constraints. In these cases, Kinesis Video Streams sends the Error acknowledgement in the response.

- Fragments that have time codes spanning longer than the maximum allowed limit and that contain more than 50 MB of data are not allowed.
- Fragments containing more than three tracks are not allowed. Each frame in every fragment must have the same track number as one of the tracks defined in the fragment header. Additionally, every fragment must contain at least one frame for each track defined in the fragment header.
- Each fragment must contain at least one frame for each track defined in the fragment metadata.
- The earliest frame timestamp in a fragment must be after the latest frame timestamp in the previous fragment.
- An MKV stream containing more than one MKV segment or containing disallowed MKV elements (like `track*`) also results in the Error acknowledgement.

Kinesis Video Streams stores each incoming fragment and related metadata in what is called a "chunk." The fragment metadata includes the following:

- The MKV headers provided at the start of the `PutMedia` request
- The following Kinesis Video Streams-specific metadata for the fragment:
  - `server_timestamp` - Timestamp when Kinesis Video Streams started receiving the fragment.
  - `producer_timestamp` - Timestamp, when the producer started recording the fragment. Kinesis Video Streams uses three pieces of information received in the request to calculate this value.
    - The fragment timecode value received in the request body along with the fragment.
    - Two request headers: `producerStartTimestamp` (when the producer started recording) and `fragmentTimeCodeType` (whether the fragment timecode in the payload is absolute or relative).

    Kinesis Video Streams then computes the `producer_timestamp` for the fragment as follows:

    If `fragmentTimeCodeType` is relative, then

    `producer_timestamp` = `producerStartTimeStamp` + fragment timecode

    If `fragmentTimeCodeType` is absolute, then

    `producer_timestamp` = fragment timecode (converted to milliseconds)
  - Unique fragment number assigned by Kinesis Video Streams.

> **Note**
>
> When you make the `GetMedia` request, Kinesis Video Streams returns a stream of these chunks. The client can process the metadata as needed.

> **Note**
>
> This operation is only available for the AWS SDK for Java. It is not supported in AWS SDKs for other languages.

> **Note**
>
> Kinesis Video Streams does not parse and validate the codec private data during ingestion and archival via the PutMedia API. KVS extracts and validates the necessary information from the codec private data for MPEG-TS and MP4 fragment packaging when consuming the stream via the HLS APIs.

> **Note**
>
> If an error is thrown after invoking a Kinesis Video Streams media API, in addition to the HTTP status code and the response body, it includes the following pieces of information:
>
> - `x-amz-ErrorType` HTTP header – contains a more specific error type in addition to what the HTTP status code provides.
> - `x-amz-RequestId` HTTP header – if you want to report an issue to AWS, the support team can better diagnose the problem if given the Request Id.
>
> Both the HTTP status code and the ErrorType header can be utilized to make programmatic decisions about whether errors are retry-able and under what conditions, as well as provide information on what actions the client programmer might need to take in order to successfully try again.
> For more information, see the **Errors** section at the bottom of this topic, as well as Common Errors.

Request Syntax

```
POST /putMedia HTTP/1.1
x-amzn-stream-name: StreamName
x-amzn-stream-arn: StreamARN
x-amzn-fragment-timecode-type: FragmentTimecodeType
x-amzn-producer-start-timestamp: ProducerStartTimestamp

Payload
```

URI Request Parameters

The request uses the following URI parameters.

FragmentTimecodeType

You pass this value as the `x-amzn-fragment-timecode-type` HTTP header.

Indicates whether timecodes in the fragments (payload, HTTP request body) are absolute or relative to `producerStartTimestamp`. Kinesis Video Streams uses this information to compute the `producer_timestamp` for the fragment received in the request, as described in the API overview.

Valid Values: `ABSOLUTE | RELATIVE`

Required: Yes

ProducerStartTimestamp

You pass this value as the `x-amzn-producer-start-timestamp` HTTP header.

This is the producer timestamp at which the producer started recording the media (not the timestamp of the specific fragments in the request).

StreamARN

You pass this value as the `x-amzn-stream-arn` HTTP header.

Amazon Resource Name (ARN) of the Kinesis video stream where you want to write the media content. If you don't specify the `streamARN`, you must specify the `streamName`.

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

StreamName

You pass this value as the `x-amzn-stream-name` HTTP header.

Name of the Kinesis video stream where you want to write the media content. If you don't specify the `streamName`, you must specify the `streamARN`.

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Request Body

The request accepts the following binary data.

Payload

The media content to write to the Kinesis video stream. In the current implementation, Kinesis Video Streams supports only the Matroska (MKV) container format with a single MKV segment. A segment can contain one or more clusters.

> **Note**
>
> Each MKV cluster maps to a Kinesis video stream fragment. Whatever cluster duration you choose becomes the fragment duration.

Response Syntax

```
HTTP/1.1 200

Payload
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The response returns the following as the HTTP body.

Payload

After Kinesis Video Streams successfully receives a `PutMedia` request, the service validates the request headers. The service then starts reading the payload and first sends an HTTP 200 response.

The service then returns a stream containing a series of JSON objects (`Acknowledgement` objects) separated by newlines. The acknowledgements are received on the same connection on which the media data is sent. There can be many acknowledgements for a `PutMedia` request. Each `Acknowledgement` consists of the following key-value pairs:

- `AckEventType` - Event type the acknowledgement represents.

- **Buffering:** Kinesis Video Streams has started receiving the fragment. Kinesis Video Streams sends the first Buffering acknowledgement when the first byte of fragment data is received.
- **Received:** Kinesis Video Streams received the entire fragment. If you did not configure the stream to persist the data, the producer can stop buffering the fragment upon receiving this acknowledgement.
- **Persisted:** Kinesis Video Streams has persisted the fragment (for example, to Amazon S3). You get this acknowledgement if you configured the stream to persist the data. After you receive this acknowledgement, the producer can stop buffering the fragment.
- **Error:** Kinesis Video Streams ran into an error while processing the fragment. You can review the error code and determine the next course of action.
- **Idle:** The `PutMedia` session is in-progress. However, Kinesis Video Streams is currently not receiving data. Kinesis Video Streams sends this acknowledgement periodically for up to 30 seconds after the last received data. If no data is received within the 30 seconds, Kinesis Video Streams closes the request.

> **Note**
>
> This acknowledgement can help a producer determine if the `PutMedia` connection is alive, even if it is not sending any data.

- `FragmentTimecode` - Fragment timecode for which acknowledgement is sent.

  The element can be missing if the `AckEventType` is **Idle**.
- `FragmentNumber` - Kinesis Video Streams-generated fragment number for which the acknowledgement is sent.
- `ErrorId` and `ErrorCode` - If the `AckEventType` is `Error`, this field provides corresponding error code. The following is the list of error IDs and their corresponding error codes and error messages:
  - 4000 - STREAM_READ_ERROR - Error reading the data stream.
  - 4001 - MAX_FRAGMENT_SIZE_REACHED - Fragment size is greater than maximum limit, 50 MB, allowed.
  - 4002 - MAX_FRAGMENT_DURATION_REACHED - Fragment duration is greater than maximum allowed limit.

- 4003 - MAX_CONNECTION_DURATION_REACHED - Connection duration is greater than maximum allowed threshold.
- 4004 - FRAGMENT_TIMECODE_LESSER_THAN_PREVIOUS - Fragment timecode is less than the timecode previous time code (within a `PutMedia` call, you cannot send fragments out of order).
- 4005 - MORE_THAN_ALLOWED_TRACKS_FOUND - More than one track is found in MKV. (deprecated)
- 4006 - INVALID_MKV_DATA - Failed to parse the input stream as valid MKV format.
- 4007 - INVALID_PRODUCER_TIMESTAMP - Invalid producer timestamp.
- 4008 - STREAM_NOT_ACTIVE - Stream no longer exists (deleted).
- 4009 - FRAGMENT_METADATA_LIMIT_REACHED - Fragment metadata limit reached. See the Limits section of the developer guide.
- 4010 - TRACK_NUMBER_MISMATCH - The track number in an MKV frame did not match the tracks in the MKV header.
- 4011 - FRAMES_MISSING_FOR_TRACK - The fragment did not contain any frames for at least one of the tracks in the MKV header.
- 4012 - INVALID_FRAGMENT_METADATA - Fragment metadata name cannot begin with the string `AWS_`.
- 4500 - KMS_KEY_ACCESS_DENIED - Access to the stream's specified KMS key is denied.
- 4501 - KMS_KEY_DISABLED - The stream's specified KMS key is disabled.
- 4502 - KMS_KEY_VALIDATION_ERROR - The stream's specified KMS key failed validation.
- 4503 - KMS_KEY_UNAVAILABLE - The stream's specified KMS key is unavailable.
- 4504 - KMS_KEY_INVALID_USAGE - Invalid usage of the stream's specified KMS key.
- 4505 - KMS_KEY_INVALID_STATE - The stream's specified KMS key is in an invalid state.
- 4506 - KMS_KEY_NOT_FOUND - The stream's specified KMS key is not found.
- 5000 - INTERNAL_ERROR - Internal service error.
- 5001 - ARCHIVAL_ERROR - Kinesis Video Streams failed to persist fragments to the data store.

> **Note**
>
> The producer, while sending the payload for a long running `PutMedia` request, should read the response for acknowledgements. A producer might receive chunks of

acknowledgements at the same time, due to buffering on an intermediate proxy server. A producer that wants to receive timely acknowledgements can send fewer fragments in each `PutMedia` request.

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

ConnectionLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client connections.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

InvalidEndpointException

Caller used wrong endpoint to write data to a stream. On receiving such an exception, the user must call `GetDataEndpoint` with `APIName` set to `PUT_MEDIA` and use the endpoint from response to invoke the next `PutMedia` call.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform an operation on the given stream, or the token has expired.

HTTP Status Code: 401

ResourceNotFoundException

Status Code: 404, The stream with the given name does not exist.

HTTP Status Code: 404

Examples

Acknowledgement Format

The format of the acknowledgement is as follows:

```
{
       Acknowledgement : {
          "EventType": enum
          "FragmentTimecode": Long,
          "FragmentNumber": Long,
          "ErrorId" : String
      }
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Amazon Kinesis Video Streams Archived Media

The following actions are supported by Amazon Kinesis Video Streams Archived Media:

- GetClip
- GetDASHStreamingSessionURL
- GetHLSStreamingSessionURL
- GetImages
- GetMediaForFragmentList
- ListFragments

GetClip

Service: Amazon Kinesis Video Streams Archived Media

Downloads an MP4 file (clip) containing the archived, on-demand media from the specified video stream over the specified time range.

Both the StreamName and the StreamARN parameters are optional, but you must specify either the StreamName or the StreamARN when invoking this API operation.

> **Note**
>
> You must first call the `GetDataEndpoint` API to get an endpoint. Then send the `GetClip` requests to this endpoint using the --endpoint-url parameter.

An Amazon Kinesis video stream has the following requirements for providing data through MP4:

- Video playback track requirements.
- Data retention must be greater than 0.
- The video track of each fragment must contain codec private data in the Advanced Video Coding (AVC) for H.264 format and HEVC for H.265 format. For more information, see MPEG-4 specification ISO/IEC 14496-15. For information about adapting stream data to a given format, see NAL Adaptation Flags.
- The audio track (if present) of each fragment must contain codec private data in the AAC format (AAC specification ISO/IEC 13818-7) or the MS Wave format.

You can monitor the amount of outgoing data by monitoring the `GetClip.OutgoingBytes` Amazon CloudWatch metric. For information about using CloudWatch to monitor Kinesis Video Streams, see Monitoring Kinesis Video Streams. For pricing information, see Amazon Kinesis Video Streams Pricing and AWS Pricing. Charges for outgoing AWS data apply.

> **Important**
>
> The codec private data (CPD) contained in each fragment contains codec-specific initialization information, such as frame rate, resolution, and encoding profile, which are necessary to properly decode the fragment. CPD changes aren't supported between the

target fragments of the resulting clip. The CPD must remain consistent through the queried media, otherwise an error will be returned.

Important

Track changes aren't supported. Tracks must remain consistent throughout the queried media. An error is returned if the fragments in the stream change from having only video to having both audio and video, or if an AAC audio track is changed to an A-Law audio track.

Request Syntax

```
POST /getClip HTTP/1.1
Content-type: application/json

{
   "ClipFragmentSelector": {
      "FragmentSelectorType": "string",
      "TimestampRange": {
         "EndTimestamp": number,
         "StartTimestamp": number
      }
   },
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ClipFragmentSelector

The time range of the requested clip and the source of the timestamps.

Type: ClipFragmentSelector object

Required: Yes

StreamARN

The Amazon Resource Name (ARN) of the stream for which to retrieve the media clip.

You must specify either the StreamName or the StreamARN.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream for which to retrieve the media clip.

You must specify either the StreamName or the StreamARN.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-Type: ContentType

Payload
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The response returns the following HTTP headers.

ContentType

The content type of the media in the requested clip.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^[a-zA-Z0-9_\.\-]+$`

The response returns the following as the HTTP body.

Payload

Traditional MP4 file that contains the media clip from the specified video stream. The output will contain the first 100 MB or the first 200 fragments from the specified start timestamp. For more information, see Kinesis Video Streams Limits.

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded a limit. Try making the call later. For information about limits, see Kinesis Video Streams Limits.

HTTP Status Code: 400

InvalidArgumentException

A specified parameter exceeds its restrictions, is not supported, or can't be used.

HTTP Status Code: 400

InvalidCodecPrivateDataException

The codec private data in at least one of the tracks of the video stream is not valid for this operation.

HTTP Status Code: 400

InvalidMediaFrameException

One or more frames in the requested clip could not be parsed based on the specified codec.

HTTP Status Code: 400

MissingCodecPrivateDataException

No codec private data was found in at least one of tracks of the video stream.

HTTP Status Code: 400

NoDataRetentionException

`GetImages` was requested for a stream that does not retain data (that is, has a `DataRetentionInHours` of 0).

HTTP Status Code: 400

NotAuthorizedException

Status Code: 403, The caller is not authorized to perform an operation on the given stream, or the token has expired.

HTTP Status Code: 401

ResourceNotFoundException

`GetImages` will throw this error when Kinesis Video Streams can't find the stream that you specified.

`GetHLSStreamingSessionURL` and `GetDASHStreamingSessionURL` throw this error if a session with a `PlaybackMode` of `ON_DEMAND` or `LIVE_REPLAY` is requested for a stream that has no fragments within the requested time range, or if a session with a `PlaybackMode` of `LIVE` is requested for a stream that has no fragments within the last 30 seconds.

HTTP Status Code: 404

UnsupportedStreamMediaTypeException

The type of the media (for example, h.264 or h.265 video or AAC or G.711 audio) could not be determined from the codec IDs of the tracks in the first fragment for a playback session. The codec ID for track 1 should be `V_MPEG/ISO/AVC` and, optionally, the codec ID for track 2 should be `A_AAC`.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetDASHStreamingSessionURL

Service: Amazon Kinesis Video Streams Archived Media

Retrieves an MPEG Dynamic Adaptive Streaming over HTTP (DASH) URL for the stream. You can then open the URL in a media player to view the stream contents.

Both the `StreamName` and the `StreamARN` parameters are optional, but you must specify either the `StreamName` or the `StreamARN` when invoking this API operation.

An Amazon Kinesis video stream has the following requirements for providing data through MPEG-DASH:

- Video playback track requirements.
- Data retention must be greater than 0.
- The video track of each fragment must contain codec private data in the Advanced Video Coding (AVC) for H.264 format and HEVC for H.265 format. For more information, see MPEG-4 specification ISO/IEC 14496-15. For information about adapting stream data to a given format, see NAL Adaptation Flags.
- The audio track (if present) of each fragment must contain codec private data in the AAC format (AAC specification ISO/IEC 13818-7) or the MS Wave format.

The following procedure shows how to use MPEG-DASH with Kinesis Video Streams:

1. Call the `GetDataEndpoint` API to get an endpoint. Then send the `GetDASHStreamingSessionURL` requests to this endpoint using the --endpoint-url parameter.
2. Retrieve the MPEG-DASH URL using `GetDASHStreamingSessionURL`. Kinesis Video Streams creates an MPEG-DASH streaming session to be used for accessing content in a stream using the MPEG-DASH protocol. `GetDASHStreamingSessionURL` returns an authenticated URL (that includes an encrypted session token) for the session's MPEG-DASH *manifest* (the root resource needed for streaming with MPEG-DASH).

> **Note**
>
> Don't share or store this token where an unauthorized entity can access it. The token provides access to the content of the stream. Safeguard the token with the same measures that you use with your AWS credentials.

The media that is made available through the manifest consists only of the requested stream, time range, and format. No other media data (such as frames outside the requested window or alternate bitrates) is made available.

3. Provide the URL (containing the encrypted session token) for the MPEG-DASH manifest to a media player that supports the MPEG-DASH protocol. Kinesis Video Streams makes the initialization fragment and media fragments available through the manifest URL. The initialization fragment contains the codec private data for the stream, and other data needed to set up the video or audio decoder and renderer. The media fragments contain encoded video frames or encoded audio samples.
4. The media player receives the authenticated URL and requests stream metadata and media data normally. When the media player requests data, it calls the following actions:
   - **GetDASHManifest:** Retrieves an MPEG DASH manifest, which contains the metadata for the media that you want to playback.
   - **GetMP4InitFragment:** Retrieves the MP4 initialization fragment. The media player typically loads the initialization fragment before loading any media fragments. This fragment contains the "`fytp`" and "`moov`" MP4 atoms, and the child atoms that are needed to initialize the media player decoder.

     The initialization fragment does not correspond to a fragment in a Kinesis video stream. It contains only the codec private data for the stream and respective track, which the media player needs to decode the media frames.
   - **GetMP4MediaFragment:** Retrieves MP4 media fragments. These fragments contain the "`moof`" and "`mdat`" MP4 atoms and their child atoms, containing the encoded fragment's media frames and their timestamps.

   > **Important**
   >
   > The codec private data (CPD) contained in each fragment contains codec-specific initialization information, such as frame rate, resolution, and encoding profile, which are necessary to properly decode the fragment. CPD changes aren't supported during a streaming session. The CPD must remain consistent through the queried media.

> **Important**
>
> Track changes aren't supported. Tracks must remain consistent throughout the queried media. Streaming will fail if the fragments in the stream change from having only video to having both audio and video, or if an AAC audio track is changed to an A-Law audio track.

Data retrieved with this action is billable. See Pricing for details.

> **Note**
>
> For restrictions that apply to MPEG-DASH sessions, see Kinesis Video Streams Limits.

You can monitor the amount of data that the media player consumes by monitoring the `GetMP4MediaFragment.OutgoingBytes` Amazon CloudWatch metric. For information about using CloudWatch to monitor Kinesis Video Streams, see Monitoring Kinesis Video Streams. For pricing information, see Amazon Kinesis Video Streams Pricing and AWS Pricing. Charges for both HLS sessions and outgoing AWS data apply.

For more information about HLS, see HTTP Live Streaming on the Apple Developer site.

> **Important**
>
> If an error is thrown after invoking a Kinesis Video Streams archived media API, in addition to the HTTP status code and the response body, it includes the following pieces of information:
>
> - `x-amz-ErrorType` HTTP header – contains a more specific error type in addition to what the HTTP status code provides.
> - `x-amz-RequestId` HTTP header – if you want to report an issue to AWS the support team can better diagnose the problem if given the Request Id.
>
> Both the HTTP status code and the ErrorType header can be utilized to make programmatic decisions about whether errors are retry-able and under what conditions, as well as

provide information on what actions the client programmer might need to take in order to successfully try again.

For more information, see the **Errors** section at the bottom of this topic, as well as Common Errors.

Request Syntax

```
POST /getDASHStreamingSessionURL HTTP/1.1
Content-type: application/json

{
   "DASHFragmentSelector": {
      "FragmentSelectorType": "string",
      "TimestampRange": {
         "EndTimestamp": number,
         "StartTimestamp": number
      }
   },
   "DisplayFragmentNumber": "string",
   "DisplayFragmentTimestamp": "string",
   "Expires": number,
   "MaxManifestFragmentResults": number,
   "PlaybackMode": "string",
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

DASHFragmentSelector

The time range of the requested fragment and the source of the timestamps.

This parameter is required if `PlaybackMode` is `ON_DEMAND` or `LIVE_REPLAY`. This parameter is optional if PlaybackMode is `LIVE`. If `PlaybackMode` is `LIVE`, the `FragmentSelectorType` can be set, but the `TimestampRange` should not be set. If `PlaybackMode` is `ON_DEMAND` or `LIVE_REPLAY`, both `FragmentSelectorType` and `TimestampRange` must be set.

Type: DASHFragmentSelector object

Required: No

DisplayFragmentNumber

Fragments are identified in the manifest file based on their sequence number in the session. If DisplayFragmentNumber is set to `ALWAYS`, the Kinesis Video Streams fragment number is added to each S element in the manifest file with the attribute name “kvs:fn”. These fragment numbers can be used for logging or for use with other APIs (e.g. `GetMedia` and `GetMediaForFragmentList`). A custom MPEG-DASH media player is necessary to leverage these this custom attribute.

The default value is `NEVER`.

Type: String

Valid Values: `ALWAYS | NEVER`

Required: No

DisplayFragmentTimestamp

Per the MPEG-DASH specification, the wall-clock time of fragments in the manifest file can be derived using attributes in the manifest itself. However, typically, MPEG-DASH compatible media players do not properly handle gaps in the media timeline. Kinesis Video Streams adjusts the media timeline in the manifest file to enable playback of media with discontinuities. Therefore, the wall-clock time derived from the manifest file may be inaccurate. If DisplayFragmentTimestamp is set to `ALWAYS`, the accurate fragment timestamp is added to each S element in the manifest file with the attribute name “kvs:ts”. A custom MPEG-DASH media player is necessary to leverage this custom attribute.

The default value is `NEVER`. When DASHFragmentSelector is `SERVER_TIMESTAMP`, the timestamps will be the server start timestamps. Similarly, when DASHFragmentSelector is `PRODUCER_TIMESTAMP`, the timestamps will be the producer start timestamps.

Type: String

Valid Values: `ALWAYS | NEVER`

Required: No

Expires

The time in seconds until the requested session expires. This value can be between 300 (5 minutes) and 43200 (12 hours).

When a session expires, no new calls to `GetDashManifest`, `GetMP4InitFragment`, or `GetMP4MediaFragment` can be made for that session.

The default is 300 (5 minutes).

Type: Integer

Valid Range: Minimum value of 300. Maximum value of 43200.

Required: No

MaxManifestFragmentResults

The maximum number of fragments that are returned in the MPEG-DASH manifest.

When the `PlaybackMode` is `LIVE`, the most recent fragments are returned up to this value. When the `PlaybackMode` is `ON_DEMAND`, the oldest fragments are returned, up to this maximum number.

When there are a higher number of fragments available in a live MPEG-DASH manifest, video players often buffer content before starting playback. Increasing the buffer size increases the playback latency, but it decreases the likelihood that rebuffering will occur during playback. We recommend that a live MPEG-DASH manifest have a minimum of 3 fragments and a maximum of 10 fragments.

The default is 5 fragments if `PlaybackMode` is `LIVE` or `LIVE_REPLAY`, and 1,000 if `PlaybackMode` is `ON_DEMAND`.

The maximum value of 1,000 fragments corresponds to more than 16 minutes of video on streams with 1-second fragments, and more than 2 1/2 hours of video on streams with 10-second fragments.

Type: Long

Valid Range: Minimum value of 1. Maximum value of 5000.

Required: No

PlaybackMode

Whether to retrieve live, live replay, or archived, on-demand data.

Features of the three types of sessions include the following:

- **`LIVE`** : For sessions of this type, the MPEG-DASH manifest is continually updated with the latest fragments as they become available. We recommend that the media player retrieve a new manifest on a one-second interval. When this type of session is played in a media player, the user interface typically displays a "live" notification, with no scrubber control for choosing the position in the playback window to display.

  Note

  In `LIVE` mode, the newest available fragments are included in an MPEG-DASH manifest, even if there is a gap between fragments (that is, if a fragment is missing). A gap like this might cause a media player to halt or cause a jump in playback. In this mode, fragments are not added to the MPEG-DASH manifest if they are older than the newest fragment in the playlist. If the missing fragment becomes available after a subsequent fragment is added to the manifest, the older fragment is not added, and the gap is not filled.

- **`LIVE_REPLAY`** : For sessions of this type, the MPEG-DASH manifest is updated similarly to how it is updated for `LIVE` mode except that it starts by including fragments from a given start time. Instead of fragments being added as they are ingested, fragments are added as the duration of the next fragment elapses. For example, if the fragments in the session are two seconds long, then a new fragment is added to the manifest every two seconds. This mode is useful to be able to start playback from when an event is detected and continue live streaming media that has not yet been ingested as of the time of the session creation. This mode is also useful to stream previously archived media without being limited by the 1,000 fragment limit in the `ON_DEMAND` mode.
- **`ON_DEMAND`** : For sessions of this type, the MPEG-DASH manifest contains all the fragments for the session, up to the number that is specified in `MaxManifestFragmentResults`. The manifest must be retrieved only once for each session. When this type of session is played in a media player, the user interface typically displays a scrubber control for choosing the position in the playback window to display.

In all playback modes, if `FragmentSelectorType` is `PRODUCER_TIMESTAMP`, and if there are multiple fragments with the same start timestamp, the fragment that has the larger fragment number (that is, the newer fragment) is included in the MPEG-DASH manifest. The other fragments are not included. Fragments that have different timestamps but have overlapping durations are still included in the MPEG-DASH manifest. This can lead to unexpected behavior in the media player.

The default is `LIVE`.

Type: String

Valid Values: `LIVE | LIVE_REPLAY | ON_DEMAND`

Required: No

StreamARN

The Amazon Resource Name (ARN) of the stream for which to retrieve the MPEG-DASH manifest URL.

You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream for which to retrieve the MPEG-DASH manifest URL.

You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "DASHStreamingSessionURL": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

DASHStreamingSessionURL

The URL (containing the session token) that a media player can use to retrieve the MPEG-DASH manifest.

Type: String

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded a limit. Try making the call later. For information about limits, see Kinesis Video Streams Limits.

HTTP Status Code: 400

InvalidArgumentException

A specified parameter exceeds its restrictions, is not supported, or can't be used.

HTTP Status Code: 400

InvalidCodecPrivateDataException

The codec private data in at least one of the tracks of the video stream is not valid for this operation.

HTTP Status Code: 400

MissingCodecPrivateDataException

No codec private data was found in at least one of tracks of the video stream.

HTTP Status Code: 400

NoDataRetentionException

`GetImages` was requested for a stream that does not retain data (that is, has a `DataRetentionInHours` of 0).

HTTP Status Code: 400

NotAuthorizedException

Status Code: 403, The caller is not authorized to perform an operation on the given stream, or the token has expired.

HTTP Status Code: 401

ResourceNotFoundException

`GetImages` will throw this error when Kinesis Video Streams can't find the stream that you specified.

`GetHLSStreamingSessionURL` and `GetDASHStreamingSessionURL` throw this error if a session with a `PlaybackMode` of `ON_DEMAND` or `LIVE_REPLAY` is requested for a stream that has no fragments within the requested time range, or if a session with a `PlaybackMode` of `LIVE` is requested for a stream that has no fragments within the last 30 seconds.

HTTP Status Code: 404

UnsupportedStreamMediaTypeException

The type of the media (for example, h.264 or h.265 video or AAC or G.711 audio) could not be determined from the codec IDs of the tracks in the first fragment for a playback session. The codec ID for track 1 should be `V_MPEG/ISO/AVC` and, optionally, the codec ID for track 2 should be `A_AAC`.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetHLSStreamingSessionURL

Service: Amazon Kinesis Video Streams Archived Media

Retrieves an HTTP Live Streaming (HLS) URL for the stream. You can then open the URL in a browser or media player to view the stream contents.

Both the `StreamName` and the `StreamARN` parameters are optional, but you must specify either the `StreamName` or the `StreamARN` when invoking this API operation.

An Amazon Kinesis video stream has the following requirements for providing data through HLS:

- Video playback track requirements.
- Data retention must be greater than 0.
- The video track of each fragment must contain codec private data in the Advanced Video Coding (AVC) for H.264 format or HEVC for H.265 format (MPEG-4 specification ISO/IEC 14496-15). For information about adapting stream data to a given format, see NAL Adaptation Flags.
- The audio track (if present) of each fragment must contain codec private data in the AAC format (AAC specification ISO/IEC 13818-7).

Kinesis Video Streams HLS sessions contain fragments in the fragmented MPEG-4 form (also called fMP4 or CMAF) or the MPEG-2 form (also called TS chunks, which the HLS specification also supports). For more information about HLS fragment types, see the HLS specification.

The following procedure shows how to use HLS with Kinesis Video Streams:

1. Call the `GetDataEndpoint` API to get an endpoint. Then send the `GetHLSStreamingSessionURL` requests to this endpoint using the --endpoint-url parameter.
2. Retrieve the HLS URL using `GetHLSStreamingSessionURL`. Kinesis Video Streams creates an HLS streaming session to be used for accessing content in a stream using the HLS protocol. `GetHLSStreamingSessionURL` returns an authenticated URL (that includes an encrypted session token) for the session's HLS *master playlist* (the root resource needed for streaming with HLS).

> **Note**
>
> Don't share or store this token where an unauthorized entity could access it. The token provides access to the content of the stream. Safeguard the token with the same measures that you would use with your AWS credentials.

The media that is made available through the playlist consists only of the requested stream, time range, and format. No other media data (such as frames outside the requested window or alternate bitrates) is made available.

3. Provide the URL (containing the encrypted session token) for the HLS master playlist to a media player that supports the HLS protocol. Kinesis Video Streams makes the HLS media playlist, initialization fragment, and media fragments available through the master playlist URL. The initialization fragment contains the codec private data for the stream, and other data needed to set up the video or audio decoder and renderer. The media fragments contain H.264-encoded video frames or AAC-encoded audio samples.
4. The media player receives the authenticated URL and requests stream metadata and media data normally. When the media player requests data, it calls the following actions:
   - **GetHLSMasterPlaylist:** Retrieves an HLS master playlist, which contains a URL for the `GetHLSMediaPlaylist` action for each track, and additional metadata for the media player, including estimated bitrate and resolution.
   - **GetHLSMediaPlaylist:** Retrieves an HLS media playlist, which contains a URL to access the MP4 initialization fragment with the `GetMP4InitFragment` action, and URLs to access the MP4 media fragments with the `GetMP4MediaFragment` actions. The HLS media playlist also contains metadata about the stream that the player needs to play it, such as whether the `PlaybackMode` is `LIVE` or `ON_DEMAND`. The HLS media playlist is typically static for sessions with a `PlaybackType` of `ON_DEMAND`. The HLS media playlist is continually updated with new fragments for sessions with a `PlaybackType` of `LIVE`. There is a distinct HLS media playlist for the video track and the audio track (if applicable) that contains MP4 media URLs for the specific track.
   - **GetMP4InitFragment:** Retrieves the MP4 initialization fragment. The media player typically loads the initialization fragment before loading any media fragments. This fragment contains the "`fytp`" and "`moov`" MP4 atoms, and the child atoms that are needed to initialize the media player decoder.

The initialization fragment does not correspond to a fragment in a Kinesis video stream. It contains only the codec private data for the stream and respective track, which the media player needs to decode the media frames.

- **GetMP4MediaFragment:** Retrieves MP4 media fragments. These fragments contain the "`moof`" and "`mdat`" MP4 atoms and their child atoms, containing the encoded fragment's media frames and their timestamps.

> **Note**
>
> The codec private data (CPD) contained in each fragment contains codec-specific initialization information, such as frame rate, resolution, and encoding profile, which are necessary for properly decoding the fragment. For both TS and MP4, CPD changes are supported during a streaming session. Therefore, the fragments in a session can have a different information in the CPD without interrupting playback. For each streaming session, only 500 CPD changes are allowed.

> **Important**
>
> Track changes aren't supported. Tracks must remain consistent throughout the queried media. Streaming will fail if the fragments in the stream change from having only video to having both audio and video, or if an AAC audio track is changed to an A-Law audio track.

  Data retrieved with this action is billable. For information, see Pricing.

- **GetTSFragment:** Retrieves MPEG TS fragments containing both initialization and media data for all tracks in the stream.

> **Note**
>
> If the `ContainerFormat` is `MPEG_TS`, this API is used instead of `GetMP4InitFragment` and `GetMP4MediaFragment` to retrieve stream media.

Data retrieved with this action is billable. For more information, see Kinesis Video Streams pricing.

A streaming session URL must not be shared between players. The service might throttle a session if multiple media players are sharing it. For connection limits, see Kinesis Video Streams Limits.

You can monitor the amount of data that the media player consumes by monitoring the `GetMP4MediaFragment.OutgoingBytes` Amazon CloudWatch metric. For information about using CloudWatch to monitor Kinesis Video Streams, see Monitoring Kinesis Video Streams. For pricing information, see Amazon Kinesis Video Streams Pricing and AWS Pricing. Charges for both HLS sessions and outgoing AWS data apply.

See the video playback examples in the documentation guide: Use the AWS CLI to retrieve an HLS streaming session URL and Example: Use HLS in HTML and JavaScript.

For more information about HLS, see HTTP Live Streaming on the Apple Developer site.

Important

If an error is thrown after invoking a Kinesis Video Streams archived media API, in addition to the HTTP status code and the response body, it includes the following pieces of information:

- `x-amz-ErrorType` HTTP header – contains a more specific error type in addition to what the HTTP status code provides.
- `x-amz-RequestId` HTTP header – if you want to report an issue to AWS, the support team can better diagnose the problem if given the Request Id.

Both the HTTP status code and the ErrorType header can be utilized to make programmatic decisions about whether errors are retry-able and under what conditions, as well as provide information on what actions the client programmer might need to take in order to successfully try again.
For more information, see the **Errors** section at the bottom of this topic, as well as Common Errors.

Request Syntax

```
POST /getHLSStreamingSessionURL HTTP/1.1
Content-type: application/json

{
   "ContainerFormat": "string",
   "DiscontinuityMode": "string",
   "DisplayFragmentTimestamp": "string",
   "Expires": number,
   "HLSFragmentSelector": {
      "FragmentSelectorType": "string",
      "TimestampRange": {
         "EndTimestamp": number,
         "StartTimestamp": number
      }
   },
   "MaxMediaPlaylistFragmentResults": number,
   "PlaybackMode": "string",
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ContainerFormat

Specifies which format should be used for packaging the media. Specifying the `FRAGMENTED_MP4` container format packages the media into MP4 fragments (fMP4 or CMAF). This is the recommended packaging because there is minimal packaging overhead. The other container format option is `MPEG_TS`. HLS has supported MPEG TS chunks since it was released and is sometimes the only supported packaging on older HLS players. MPEG TS typically has a 5-25 percent packaging overhead. This means MPEG TS typically requires 5-25 percent more bandwidth and cost than fMP4.

The default is `FRAGMENTED_MP4`.

Type: String

Valid Values: FRAGMENTED_MP4 | MPEG_TS

Required: No

DiscontinuityMode

Specifies when flags marking discontinuities between fragments are added to the media playlists.

Media players typically build a timeline of media content to play, based on the timestamps of each fragment. This means that if there is any overlap or gap between fragments (as is typical if HLSFragmentSelector is set to SERVER_TIMESTAMP), the media player timeline will also have small gaps between fragments in some places, and will overwrite frames in other places. Gaps in the media player timeline can cause playback to stall and overlaps can cause playback to be jittery. When there are discontinuity flags between fragments, the media player is expected to reset the timeline, resulting in the next fragment being played immediately after the previous fragment.

The following modes are supported:

- ALWAYS: a discontinuity marker is placed between every fragment in the HLS media playlist. It is recommended to use a value of ALWAYS if the fragment timestamps are not accurate.
- NEVER: no discontinuity markers are placed anywhere. It is recommended to use a value of NEVER to ensure the media player timeline most accurately maps to the producer timestamps.
- ON_DISCONTINUITY: a discontinuity marker is placed between fragments that have a gap or overlap of more than 50 milliseconds. For most playback scenarios, it is recommended to use a value of ON_DISCONTINUITY so that the media player timeline is only reset when there is a significant issue with the media timeline (e.g. a missing fragment).

The default is ALWAYS when HLSFragmentSelector is set to SERVER_TIMESTAMP, and NEVER when it is set to PRODUCER_TIMESTAMP.

Type: String

Valid Values: ALWAYS | NEVER | ON_DISCONTINUITY

Required: No

DisplayFragmentTimestamp

Specifies when the fragment start timestamps should be included in the HLS media playlist. Typically, media players report the playhead position as a time relative to the start of the first fragment in the playback session. However, when the start timestamps are included in the HLS media playlist, some media players might report the current playhead as an absolute time based on the fragment timestamps. This can be useful for creating a playback experience that shows viewers the wall-clock time of the media.

The default is `NEVER`. When HLSFragmentSelector is `SERVER_TIMESTAMP`, the timestamps will be the server start timestamps. Similarly, when HLSFragmentSelector is `PRODUCER_TIMESTAMP`, the timestamps will be the producer start timestamps.

Type: String

Valid Values: `ALWAYS | NEVER`

Required: No

Expires

The time in seconds until the requested session expires. This value can be between 300 (5 minutes) and 43200 (12 hours).

When a session expires, no new calls to `GetHLSMasterPlaylist`, `GetHLSMediaPlaylist`, `GetMP4InitFragment`, `GetMP4MediaFragment`, or `GetTSFragment` can be made for that session.

The default is 300 (5 minutes).

Type: Integer

Valid Range: Minimum value of 300. Maximum value of 43200.

Required: No

HLSFragmentSelector

The time range of the requested fragment and the source of the timestamps.

This parameter is required if `PlaybackMode` is `ON_DEMAND` or `LIVE_REPLAY`. This parameter is optional if PlaybackMode is `LIVE`. If `PlaybackMode` is `LIVE`, the `FragmentSelectorType` can be set, but the `TimestampRange` should not be set. If `PlaybackMode` is `ON_DEMAND` or `LIVE_REPLAY`, both `FragmentSelectorType` and `TimestampRange` must be set.

Type: HLSFragmentSelector object

Required: No

MaxMediaPlaylistFragmentResults

The maximum number of fragments that are returned in the HLS media playlists.

When the `PlaybackMode` is `LIVE`, the most recent fragments are returned up to this value. When the `PlaybackMode` is `ON_DEMAND`, the oldest fragments are returned, up to this maximum number.

When there are a higher number of fragments available in a live HLS media playlist, video players often buffer content before starting playback. Increasing the buffer size increases the playback latency, but it decreases the likelihood that rebuffering will occur during playback. We recommend that a live HLS media playlist have a minimum of 3 fragments and a maximum of 10 fragments.

The default is 5 fragments if `PlaybackMode` is `LIVE` or `LIVE_REPLAY`, and 1,000 if `PlaybackMode` is `ON_DEMAND`.

The maximum value of 5,000 fragments corresponds to more than 80 minutes of video on streams with 1-second fragments, and more than 13 hours of video on streams with 10-second fragments.

Type: Long

Valid Range: Minimum value of 1. Maximum value of 5000.

Required: No

PlaybackMode

Whether to retrieve live, live replay, or archived, on-demand data.

Features of the three types of sessions include the following:

- **`LIVE`** : For sessions of this type, the HLS media playlist is continually updated with the latest fragments as they become available. We recommend that the media player retrieve a new playlist on a one-second interval. When this type of session is played in a media player, the user interface typically displays a "live" notification, with no scrubber control for choosing the position in the playback window to display.

> **Note**
>
> In LIVE mode, the newest available fragments are included in an HLS media playlist, even if there is a gap between fragments (that is, if a fragment is missing). A gap like this might cause a media player to halt or cause a jump in playback. In this mode, fragments are not added to the HLS media playlist if they are older than the newest fragment in the playlist. If the missing fragment becomes available after a subsequent fragment is added to the playlist, the older fragment is not added, and the gap is not filled.

- **`LIVE_REPLAY`** : For sessions of this type, the HLS media playlist is updated similarly to how it is updated for `LIVE` mode except that it starts by including fragments from a given start time. Instead of fragments being added as they are ingested, fragments are added as the duration of the next fragment elapses. For example, if the fragments in the session are two seconds long, then a new fragment is added to the media playlist every two seconds. This mode is useful to be able to start playback from when an event is detected and continue live streaming media that has not yet been ingested as of the time of the session creation. This mode is also useful to stream previously archived media without being limited by the 1,000 fragment limit in the `ON_DEMAND` mode.
- **`ON_DEMAND`** : For sessions of this type, the HLS media playlist contains all the fragments for the session, up to the number that is specified in `MaxMediaPlaylistFragmentResults`. The playlist must be retrieved only once for each session. When this type of session is played in a media player, the user interface typically displays a scrubber control for choosing the position in the playback window to display.

In all playback modes, if `FragmentSelectorType` is `PRODUCER_TIMESTAMP`, and if there are multiple fragments with the same start timestamp, the fragment that has the largest fragment number (that is, the newest fragment) is included in the HLS media playlist. The other fragments are not included. Fragments that have different timestamps but have overlapping durations are still included in the HLS media playlist. This can lead to unexpected behavior in the media player.

The default is `LIVE`.

Type: String

Valid Values: `LIVE | LIVE_REPLAY | ON_DEMAND`

Required: No

StreamARN

The Amazon Resource Name (ARN) of the stream for which to retrieve the HLS master playlist URL.

You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream for which to retrieve the HLS master playlist URL.

You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "HLSStreamingSessionURL": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

HLSStreamingSessionURL

The URL (containing the session token) that a media player can use to retrieve the HLS master playlist.

Type: String

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded a limit. Try making the call later. For information about limits, see Kinesis Video Streams Limits.

HTTP Status Code: 400

InvalidArgumentException

A specified parameter exceeds its restrictions, is not supported, or can't be used.

HTTP Status Code: 400

InvalidCodecPrivateDataException

The codec private data in at least one of the tracks of the video stream is not valid for this operation.

HTTP Status Code: 400

MissingCodecPrivateDataException

No codec private data was found in at least one of tracks of the video stream.

HTTP Status Code: 400

NoDataRetentionException

`GetImages` was requested for a stream that does not retain data (that is, has a `DataRetentionInHours` of 0).

HTTP Status Code: 400

NotAuthorizedException

Status Code: 403, The caller is not authorized to perform an operation on the given stream, or the token has expired.

HTTP Status Code: 401

ResourceNotFoundException

`GetImages` will throw this error when Kinesis Video Streams can't find the stream that you specified.

`GetHLSStreamingSessionURL` and `GetDASHStreamingSessionURL` throw this error if a session with a `PlaybackMode` of `ON_DEMAND` or `LIVE_REPLAY` is requested for a stream that has no fragments within the requested time range, or if a session with a `PlaybackMode` of `LIVE` is requested for a stream that has no fragments within the last 30 seconds.

HTTP Status Code: 404

UnsupportedStreamMediaTypeException

The type of the media (for example, h.264 or h.265 video or AAC or G.711 audio) could not be determined from the codec IDs of the tracks in the first fragment for a playback session. The codec ID for track 1 should be `V_MPEG/ISO/AVC` and, optionally, the codec ID for track 2 should be `A_AAC`.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3

- AWS SDK for Python
- AWS SDK for Ruby V3

GetImages

Service: Amazon Kinesis Video Streams Archived Media

Retrieves a list of images corresponding to each timestamp for a given time range, sampling interval, and image format configuration.

> **Note**
>
> You must first call the `GetDataEndpoint` API to get an endpoint. Then send the `GetImages` requests to this endpoint using the --endpoint-url parameter.

Video playback track requirements.

Request Syntax

```
POST /getImages HTTP/1.1
Content-type: application/json

{
   "EndTimestamp": number,
   "Format": "string",
   "FormatConfig": {
      "string" : "string"
   },
   "HeightPixels": number,
   "ImageSelectorType": "string",
   "MaxResults": number,
   "NextToken": "string",
   "SamplingInterval": number,
   "StartTimestamp": number,
   "StreamARN": "string",
   "StreamName": "string",
   "WidthPixels": number
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

EndTimestamp

The end timestamp for the range of images to be generated. If the time range between `StartTimestamp` and `EndTimestamp` is more than 300 seconds above `StartTimestamp`, you will receive an `IllegalArgumentException`.

Type: Timestamp

Required: Yes

Format

The format that will be used to encode the image.

Type: String

Valid Values: `JPEG | PNG`

Required: Yes

FormatConfig

The list of a key-value pair structure that contains extra parameters that can be applied when the image is generated. The `FormatConfig` key is the `JPEGQuality`, which indicates the JPEG quality key to be used to generate the image. The `FormatConfig` value accepts ints from 1 to 100. If the value is 1, the image will be generated with less quality and the best compression. If the value is 100, the image will be generated with the best quality and less compression. If no value is provided, the default value of the `JPEGQuality` key will be set to 80.

Type: String to string map

Map Entries: Maximum number of 1 item.

Valid Keys: `JPEGQuality`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `^[a-zA-Z_0-9]+`

Required: No

HeightPixels

The height of the output image that is used in conjunction with the `WidthPixels` parameter. When both `HeightPixels` and `WidthPixels` parameters are provided, the image will be stretched to fit the specified aspect ratio. If only the `HeightPixels` parameter is provided, its original aspect ratio will be used to calculate the `WidthPixels` ratio. If neither parameter is provided, the original image size will be returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 2160.

Required: No

ImageSelectorType

The origin of the Server or Producer timestamps to use to generate the images.

Type: String

Valid Values: `PRODUCER_TIMESTAMP | SERVER_TIMESTAMP`

Required: Yes

MaxResults

The maximum number of images to be returned by the API.

> **Note**
>
> The default limit is 25 images per API response. Providing a `MaxResults` greater than this value will result in a page size of 25. Any additional results will be paginated.

Type: Long

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

NextToken

A token that specifies where to start paginating the next set of Images. This is the `GetImages:NextToken` from a previously truncated response.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 4096.

Pattern: `[a-zA-Z0-9+/]+={0,2}`

Required: No

SamplingInterval

The time interval in milliseconds (ms) at which the images need to be generated from the stream. The minimum value that can be provided is 200 ms (5 images per second). If the timestamp range is less than the sampling interval, the image from the `startTimestamp` will be returned if available.

Type: Integer

Required: Yes

StartTimestamp

The starting point from which the images should be generated. This `StartTimestamp` must be within an inclusive range of timestamps for an image to be returned.

Type: Timestamp

Required: Yes

StreamARN

The Amazon Resource Name (ARN) of the stream from which to retrieve the images. You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream from which to retrieve the images. You must specify either the `StreamName` or the `StreamARN`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

WidthPixels

The width of the output image that is used in conjunction with the `HeightPixels` parameter. When both `WidthPixels` and `HeightPixels` parameters are provided, the image will be stretched to fit the specified aspect ratio. If only the `WidthPixels` parameter is provided or if only the `HeightPixels` is provided, a `ValidationException` will be thrown. If neither parameter is provided, the original image size from the stream will be returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 3840.

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Images": [
      {
         "Error": "string",
         "ImageContent": "string",
         "TimeStamp": number
      }
   ],
   "NextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Images

The list of images generated from the video stream. If there is no media available for the given timestamp, the `NO_MEDIA` error will be listed in the output. If an error occurs while the image is being generated, the `MEDIA_ERROR` will be listed in the output as the cause of the missing image.

Type: Array of Image objects

NextToken

The encrypted token that was used in the request to get more images.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 4096.

Pattern: `[a-zA-Z0-9+/]+={0,2}`

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded a limit. Try making the call later. For information about limits, see Kinesis Video Streams Limits.

HTTP Status Code: 400

InvalidArgumentException

A specified parameter exceeds its restrictions, is not supported, or can't be used.

HTTP Status Code: 400

NotAuthorizedException

Status Code: 403, The caller is not authorized to perform an operation on the given stream, or the token has expired.

HTTP Status Code: 401

ResourceNotFoundException

`GetImages` will throw this error when Kinesis Video Streams can't find the stream that you specified.

`GetHLSStreamingSessionURL` and `GetDASHStreamingSessionURL` throw this error if a session with a `PlaybackMode` of `ON_DEMAND` or `LIVE_REPLAY` is requested for a stream that has no fragments within the requested time range, or if a session with a `PlaybackMode` of `LIVE` is requested for a stream that has no fragments within the last 30 seconds.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetMediaForFragmentList

Service: Amazon Kinesis Video Streams Archived Media

Gets media for a list of fragments (specified by fragment number) from the archived data in an Amazon Kinesis video stream.

> **Note**
>
> You must first call the `GetDataEndpoint` API to get an endpoint. Then send the `GetMediaForFragmentList` requests to this endpoint using the [--endpoint-url parameter](#).

For limits, see [Kinesis Video Streams Limits](#).

> **Important**
>
> If an error is thrown after invoking a Kinesis Video Streams archived media API, in addition to the HTTP status code and the response body, it includes the following pieces of information:
>
> - `x-amz-ErrorType` HTTP header – contains a more specific error type in addition to what the HTTP status code provides.
> - `x-amz-RequestId` HTTP header – if you want to report an issue to AWS, the support team can better diagnose the problem if given the Request Id.
>
> Both the HTTP status code and the ErrorType header can be utilized to make programmatic decisions about whether errors are retry-able and under what conditions, as well as provide information on what actions the client programmer might need to take in order to successfully try again.
> For more information, see the **Errors** section at the bottom of this topic, as well as [Common Errors](#).

Request Syntax

```
POST /getMediaForFragmentList HTTP/1.1
Content-type: application/json
```

```
{
   "Fragments": [ "string" ],
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

Fragments

A list of the numbers of fragments for which to retrieve media. You retrieve these values with ListFragments.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 1000 items.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^[0-9]+$`

Required: Yes

StreamARN

The Amazon Resource Name (ARN) of the stream from which to retrieve fragment media. Specify either this parameter or the `StreamName` parameter.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream from which to retrieve fragment media. Specify either this parameter or the `StreamARN` parameter.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-Type: ContentType

Payload
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The response returns the following HTTP headers.

ContentType

The content type of the requested media.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^[a-zA-Z0-9_\.\-]+$`

The response returns the following as the HTTP body.

Payload

The payload that Kinesis Video Streams returns is a sequence of chunks from the specified stream. For information about the chunks, see PutMedia. The chunks that Kinesis Video Streams returns in the `GetMediaForFragmentList` call also include the following additional Matroska (MKV) tags:

- AWS_KINESISVIDEO_FRAGMENT_NUMBER - Fragment number returned in the chunk.
- AWS_KINESISVIDEO_SERVER_SIDE_TIMESTAMP - Server-side timestamp of the fragment.
- AWS_KINESISVIDEO_PRODUCER_SIDE_TIMESTAMP - Producer-side timestamp of the fragment.

The following tags will be included if an exception occurs:

- AWS_KINESISVIDEO_FRAGMENT_NUMBER - The number of the fragment that threw the exception.
- AWS_KINESISVIDEO_EXCEPTION_ERROR_CODE - The integer code of the error.
- AWS_KINESISVIDEO_EXCEPTION_MESSAGE - A text description of the exception.

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded a limit. Try making the call later. For information about limits, see Kinesis Video Streams Limits.

HTTP Status Code: 400

InvalidArgumentException

A specified parameter exceeds its restrictions, is not supported, or can't be used.

HTTP Status Code: 400

NotAuthorizedException

Status Code: 403, The caller is not authorized to perform an operation on the given stream, or the token has expired.

HTTP Status Code: 401

ResourceNotFoundException

`GetImages` will throw this error when Kinesis Video Streams can't find the stream that you specified.

`GetHLSStreamingSessionURL` and `GetDASHStreamingSessionURL` throw this error if a session with a `PlaybackMode` of `ON_DEMAND` or `LIVE_REPLAY` is requested for a stream that has no fragments within the requested time range, or if a session with a `PlaybackMode` of `LIVE` is requested for a stream that has no fragments within the last 30 seconds.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListFragments

Service: Amazon Kinesis Video Streams Archived Media

Returns a list of Fragment objects from the specified stream and timestamp range within the archived data.

Listing fragments is eventually consistent. This means that even if the producer receives an acknowledgment that a fragment is persisted, the result might not be returned immediately from a request to `ListFragments`. However, results are typically available in less than one second.

> **Note**
>
> You must first call the `GetDataEndpoint` API to get an endpoint. Then send the `ListFragments` requests to this endpoint using the --endpoint-url parameter.

> **Important**
>
> If an error is thrown after invoking a Kinesis Video Streams archived media API, in addition to the HTTP status code and the response body, it includes the following pieces of information:
>
> - `x-amz-ErrorType` HTTP header – contains a more specific error type in addition to what the HTTP status code provides.
> - `x-amz-RequestId` HTTP header – if you want to report an issue to AWS, the support team can better diagnose the problem if given the Request Id.
>
> Both the HTTP status code and the ErrorType header can be utilized to make programmatic decisions about whether errors are retry-able and under what conditions, as well as provide information on what actions the client programmer might need to take in order to successfully try again.
> For more information, see the **Errors** section at the bottom of this topic, as well as Common Errors.

Request Syntax

```
POST /listFragments HTTP/1.1
```

```
Content-type: application/json

{
   "FragmentSelector": {
      "FragmentSelectorType": "string",
      "TimestampRange": {
         "EndTimestamp": number,
         "StartTimestamp": number
      }
   },
   "MaxResults": number,
   "NextToken": "string",
   "StreamARN": "string",
   "StreamName": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

FragmentSelector

Describes the timestamp range and timestamp origin for the range of fragments to return.

> **Note**
> This is only required when the `NextToken` isn't passed in the API.

Type: FragmentSelector object

Required: No

MaxResults

The total number of fragments to return. If the total number of fragments available is more than the value specified in `max-results`, then a ListFragments:NextToken is provided in the output that you can use to resume pagination.

The default value is 100.

Type: Long

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken

A token to specify where to start paginating. This is the ListFragments:NextToken from a previously truncated response.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 4096.

Pattern: `[a-zA-Z0-9+/]+={0,2}`

Required: No

StreamARN

The Amazon Resource Name (ARN) of the stream from which to retrieve a fragment list. Specify either this parameter or the `StreamName` parameter.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream from which to retrieve a fragment list. Specify either this parameter or the `StreamARN` parameter.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Fragments": [
      {
         "FragmentLengthInMilliseconds": number,
         "FragmentNumber": "string",
         "FragmentSizeInBytes": number,
         "ProducerTimestamp": number,
         "ServerTimestamp": number
      }
   ],
   "NextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Fragments

A list of archived Fragment objects from the stream that meet the selector criteria. Results are in no specific order, even across pages.

If there are no fragments in the stream that meet the selector criteria, an empty list is returned.

Type: Array of Fragment objects

NextToken

If the returned list is truncated, the operation returns this token to use to retrieve the next page of results. This value is `null` when there are no more results to return.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 4096.

Pattern: `[a-zA-Z0-9+/]+={0,2}`

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded a limit. Try making the call later. For information about limits, see Kinesis Video Streams Limits.

HTTP Status Code: 400

InvalidArgumentException

A specified parameter exceeds its restrictions, is not supported, or can't be used.

HTTP Status Code: 400

NotAuthorizedException

Status Code: 403, The caller is not authorized to perform an operation on the given stream, or the token has expired.

HTTP Status Code: 401

ResourceNotFoundException

`GetImages` will throw this error when Kinesis Video Streams can't find the stream that you specified.

`GetHLSStreamingSessionURL` and `GetDASHStreamingSessionURL` throw this error if a session with a `PlaybackMode` of `ON_DEMAND` or `LIVE_REPLAY` is requested for a stream that has no fragments within the requested time range, or if a session with a `PlaybackMode` of `LIVE` is requested for a stream that has no fragments within the last 30 seconds.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Amazon Kinesis Video Signaling Channels

The following actions are supported by Amazon Kinesis Video Signaling Channels:

- GetIceServerConfig
- SendAlexaOfferToMaster

GetIceServerConfig

Service: Amazon Kinesis Video Signaling Channels

Note: Before using this API, you must call the `GetSignalingChannelEndpoint` API to request the HTTPS endpoint. You then specify the endpoint and region in your `GetIceServerConfig` API request.

Gets the Interactive Connectivity Establishment (ICE) server configuration information, including URIs, user name, and password which can be used to configure the WebRTC connection. The ICE component uses this configuration information to set up the WebRTC connection, including authenticating with the Traversal Using Relays around NAT (TURN) relay server.

TURN is a protocol that is used to improve the connectivity of peer-to-peer applications. By providing a cloud-based relay service, TURN ensures that a connection can be established even when one or more peers are incapable of a direct peer-to-peer connection. For more information, see A REST API For Access To TURN Services.

You can invoke this API to establish a fallback mechanism in case either of the peers is unable to establish a direct peer-to-peer connection over a signaling channel. You must specify the Amazon Resource Name (ARN) of your signaling channel in order to invoke this API.

Request Syntax

```
POST /v1/get-ice-server-config HTTP/1.1
Content-type: application/json

{
   "ChannelARN": "string",
   "ClientId": "string",
   "Service": "string",
   "Username": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ChannelARN

The ARN of the signaling channel to be used for the peer-to-peer connection between configured peers.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: Yes

ClientId

Unique identifier for the viewer. Must be unique within the signaling channel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Service

Specifies the desired service. Currently, `TURN` is the only valid value.

Type: String

Valid Values: `TURN`

Required: No

Username

An optional user ID to be associated with the credentials.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "IceServerList": [ 
      { 
         "Password": "string",
         "Ttl": number,
         "Uris": [ "string" ],
         "Username": "string"
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

IceServerList

The list of ICE server information objects.

Type: Array of IceServer objects

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Your request was throttled because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

InvalidClientException

The specified client is invalid.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform this operation.

HTTP Status Code: 401

ResourceNotFoundException

The specified resource is not found.

HTTP Status Code: 404

SessionExpiredException

If the client session is expired. Once the client is connected, the session is valid for 45 minutes. Client should reconnect to the channel to continue sending/receiving messages.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

SendAlexaOfferToMaster

Service: Amazon Kinesis Video Signaling Channels

> **Note**
>
> Before using this API, you must call the `GetSignalingChannelEndpoint` API to get an endpoint. You then specify the endpoint and region in your `SendAlexaOfferToMaster` API request.

This API allows you to connect WebRTC-enabled devices with Alexa display devices. When invoked, it sends the Alexa Session Description Protocol (SDP) offer to the master peer. The offer is delivered as soon as the master is connected to the specified signaling channel. This API returns the SDP answer from the connected master. If the master is not connected to the signaling channel, redelivery requests are made until the message expires.

Request Syntax

```
POST /v1/send-alexa-offer-to-master HTTP/1.1
Content-type: application/json

{
   "ChannelARN": "string",
   "MessagePayload": "string",
   "SenderClientId": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ChannelARN

The Amazon Resource Name (ARN) of the signaling channel by which Alexa and the master peer communicate.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: Yes

MessagePayload

The base64-encoded SDP offer content.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 10000.

Pattern: `[a-zA-Z0-9+/=]+`

Required: Yes

SenderClientId

The unique identifier for the sender client.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Answer": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Answer

The base64-encoded SDP answer content.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 10000.

Errors

For information about the errors that are common to all actions, see Common Errors.

ClientLimitExceededException

Your request was throttled because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

NotAuthorizedException

The caller is not authorized to perform this operation.

HTTP Status Code: 401

ResourceNotFoundException

The specified resource is not found.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Amazon Kinesis Video WebRTC Storage

The following actions are supported by Amazon Kinesis Video WebRTC Storage:

- JoinStorageSession

JoinStorageSession

Service: Amazon Kinesis Video WebRTC Storage

> **Note**
>
> Before using this API, you must call the `GetSignalingChannelEndpoint` API to request the WEBRTC endpoint. You then specify the endpoint and region in your `JoinStorageSession` API request.

Join the ongoing one way-video and/or multi-way audio WebRTC session as a video producing device for an input channel. If there's no existing session for the channel, a new streaming session needs to be created, and the Amazon Resource Name (ARN) of the signaling channel must be provided.

Currently for the `SINGLE_MASTER` type, a video producing device is able to ingest both audio and video media into a stream. Only video producing devices can join the session and record media.

> **Important**
>
> Both audio and video tracks are currently required for WebRTC ingestion.
> Current requirements:
>
> - Video track: H.264
> - Audio track: Opus

The resulting ingested video in the Kinesis video stream will have the following parameters: H.264 video and AAC audio.

Once a master participant has negotiated a connection through WebRTC, the ingested media session will be stored in the Kinesis video stream. Multiple viewers are then able to play back real-time media through our Playback APIs.

You can also use existing Kinesis Video Streams features like `HLS` or `DASH` playback, image generation via GetImages, and more with ingested WebRTC media.

> **Note**
>
> S3 image delivery and notifications are not currently supported.

> **Note**
>
> Assume that only one video producing device client can be associated with a session for the channel. If more than one client joins the session of a specific channel as a video producing device, the most recent client request takes precedence.

Additional information

- **Idempotent** - This API is not idempotent.
- **Retry behavior** - This is counted as a new API call.
- **Concurrent calls** - Concurrent calls are allowed. An offer is sent once per each call.

Request Syntax

```
POST /joinStorageSession HTTP/1.1
Content-type: application/json

{
   "channelArn": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

channelArn

The Amazon Resource Name (ARN) of the signaling channel.

Type: String

Pattern: `^arn:(aws[a-zA-Z-]*):kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+$`

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

AccessDeniedException

You do not have required permissions to perform this operation.

HTTP Status Code: 403

ClientLimitExceededException

Kinesis Video Streams has throttled the request because you have exceeded the limit of allowed client calls. Try making the call later.

HTTP Status Code: 400

InvalidArgumentException

The value for this input parameter is invalid.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource is not found.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Data Types

The following data types are supported by Amazon Kinesis Video Streams:

- ChannelInfo
- ChannelNameCondition
- DeletionConfig
- EdgeAgentStatus
- EdgeConfig
- ImageGenerationConfiguration
- ImageGenerationDestinationConfig
- LastRecorderStatus
- LastUploaderStatus
- ListEdgeAgentConfigurationsEdgeConfig
- LocalSizeConfig
- MappedResourceConfigurationListItem
- MediaSourceConfig
- MediaStorageConfiguration

- NotificationConfiguration
- NotificationDestinationConfig
- RecorderConfig
- ResourceEndpointListItem
- ScheduleConfig
- SingleMasterChannelEndpointConfiguration
- SingleMasterConfiguration
- StreamInfo
- StreamNameCondition
- Tag
- UploaderConfig

The following data types are supported by Amazon Kinesis Video Streams Media:

- StartSelector

The following data types are supported by Amazon Kinesis Video Streams Archived Media:

- ClipFragmentSelector
- ClipTimestampRange
- DASHFragmentSelector
- DASHTimestampRange
- Fragment
- FragmentSelector
- HLSFragmentSelector
- HLSTimestampRange
- Image
- TimestampRange

The following data types are supported by Amazon Kinesis Video Signaling Channels:

- IceServer

The following data types are supported by Amazon Kinesis Video WebRTC Storage:

Amazon Kinesis Video Streams

The following data types are supported by Amazon Kinesis Video Streams:

- ChannelInfo
- ChannelNameCondition
- DeletionConfig
- EdgeAgentStatus
- EdgeConfig
- ImageGenerationConfiguration
- ImageGenerationDestinationConfig
- LastRecorderStatus
- LastUploaderStatus
- ListEdgeAgentConfigurationsEdgeConfig
- LocalSizeConfig
- MappedResourceConfigurationListItem
- MediaSourceConfig
- MediaStorageConfiguration
- NotificationConfiguration
- NotificationDestinationConfig
- RecorderConfig
- ResourceEndpointListItem
- ScheduleConfig
- SingleMasterChannelEndpointConfiguration
- SingleMasterConfiguration
- StreamInfo
- StreamNameCondition
- Tag
- UploaderConfig

ChannelInfo

Service: Amazon Kinesis Video Streams

A structure that encapsulates a signaling channel's metadata and properties.

Contents

ChannelARN

The Amazon Resource Name (ARN) of the signaling channel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

ChannelName

The name of the signaling channel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

ChannelStatus

Current status of the signaling channel.

Type: String

Valid Values: `CREATING | ACTIVE | UPDATING | DELETING`

Required: No

ChannelType

The type of the signaling channel.

Type: String

Valid Values: `SINGLE_MASTER | FULL_MESH`

Required: No

CreationTime

The time at which the signaling channel was created.

Type: Timestamp

Required: No

SingleMasterConfiguration

A structure that contains the configuration for the `SINGLE_MASTER` channel type.

Type: SingleMasterConfiguration object

Required: No

Version

The current version of the signaling channel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `[a-zA-Z0-9]+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ChannelNameCondition

Service: Amazon Kinesis Video Streams

An optional input parameter for the `ListSignalingChannels` API. When this parameter is specified while invoking `ListSignalingChannels`, the API returns only the channels that satisfy a condition specified in `ChannelNameCondition`.

Contents

ComparisonOperator

A comparison operator. Currently, you can only specify the `BEGINS_WITH` operator, which finds signaling channels whose names begin with a given prefix.

Type: String

Valid Values: `BEGINS_WITH`

Required: No

ComparisonValue

A value to compare.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DeletionConfig

Service: Amazon Kinesis Video Streams

The configuration details required to delete the connection of the stream from the Edge Agent.

Contents

DeleteAfterUpload

The `boolean` value used to indicate whether or not you want to mark the media for deletion, once it has been uploaded to the Kinesis Video Stream cloud. The media files can be deleted if any of the deletion configuration values are set to `true`, such as when the limit for the `EdgeRetentionInHours`, or the `MaxLocalMediaSizeInMB`, has been reached.

Since the default value is set to `true`, configure the uploader schedule such that the media files are not being deleted before they are initially uploaded to the AWS cloud.

Type: Boolean

Required: No

EdgeRetentionInHours

The number of hours that you want to retain the data in the stream on the Edge Agent. The default value of the retention time is 720 hours, which translates to 30 days.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 720.

Required: No

LocalSizeConfig

The value of the local size required in order to delete the edge configuration.

Type: LocalSizeConfig object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

EdgeAgentStatus

Service: Amazon Kinesis Video Streams

An object that contains the latest status details for an edge agent's recorder and uploader jobs. Use this information to determine the current health of an edge agent.

Contents

LastRecorderStatus

The latest status of a stream's edge recording job.

Type: LastRecorderStatus object

Required: No

LastUploaderStatus

The latest status of a stream's edge to cloud uploader job.

Type: LastUploaderStatus object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

EdgeConfig

Service: Amazon Kinesis Video Streams

A description of the stream's edge configuration that will be used to sync with the Edge Agent IoT Greengrass component. The Edge Agent component will run on an IoT Hub Device setup at your premise.

Contents

HubDeviceArn

The "**Internet of Things (IoT) Thing**" Arn of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:iot:[a-z0-9-]+:[0-9]+:thing/[a-zA-Z0-9_.-]+`

Required: Yes

RecorderConfig

The recorder configuration consists of the local `MediaSourceConfig` details, that are used as credentials to access the local media files streamed on the camera.

Type: RecorderConfig object

Required: Yes

DeletionConfig

The deletion configuration is made up of the retention time (`EdgeRetentionInHours`) and local size configuration (`LocalSizeConfig`) details that are used to make the deletion.

Type: DeletionConfig object

Required: No

UploaderConfig

The uploader configuration contains the `ScheduleExpression` details that are used to schedule upload jobs for the recorded media files from the Edge Agent to a Kinesis Video Stream.

Type: UploaderConfig object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ImageGenerationConfiguration

Service: Amazon Kinesis Video Streams

The structure that contains the information required for the KVS images delivery. If null, the configuration will be deleted from the stream.

Contents

DestinationConfig

The structure that contains the information required to deliver images to a customer.

Type: ImageGenerationDestinationConfig object

Required: Yes

Format

The accepted image format.

Type: String

Valid Values: `JPEG | PNG`

Required: Yes

ImageSelectorType

The origin of the Server or Producer timestamps to use to generate the images.

Type: String

Valid Values: `SERVER_TIMESTAMP | PRODUCER_TIMESTAMP`

Required: Yes

SamplingInterval

The time interval in milliseconds (ms) at which the images need to be generated from the stream. The minimum value that can be provided is 200 ms. If the timestamp range is less than the sampling interval, the Image from the `StartTimestamp` will be returned if available.

Type: Integer

Valid Range: Minimum value of 3000. Maximum value of 20000.

Required: Yes

Status

Indicates whether the `ContinuousImageGenerationConfigurations` API is enabled or disabled.

Type: String

Valid Values: `ENABLED | DISABLED`

Required: Yes

FormatConfig

The list of a key-value pair structure that contains extra parameters that can be applied when the image is generated. The `FormatConfig` key is the `JPEGQuality`, which indicates the JPEG quality key to be used to generate the image. The `FormatConfig` value accepts ints from 1 to 100. If the value is 1, the image will be generated with less quality and the best compression. If the value is 100, the image will be generated with the best quality and less compression. If no value is provided, the default value of the `JPEGQuality` key will be set to 80.

Type: String to string map

Map Entries: Maximum number of 1 item.

Valid Keys: `JPEGQuality`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `^[a-zA-Z_0-9]+`

Required: No

HeightPixels

The height of the output image that is used in conjunction with the `WidthPixels` parameter. When both `HeightPixels` and `WidthPixels` parameters are provided, the image will be stretched to fit the specified aspect ratio. If only the `HeightPixels` parameter is provided, its original aspect ratio will be used to calculate the `WidthPixels` ratio. If neither parameter is provided, the original image size will be returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 2160.

Required: No

WidthPixels

The width of the output image that is used in conjunction with the `HeightPixels` parameter. When both `WidthPixels` and `HeightPixels` parameters are provided, the image will be stretched to fit the specified aspect ratio. If only the `WidthPixels` parameter is provided, its original aspect ratio will be used to calculate the `HeightPixels` ratio. If neither parameter is provided, the original image size will be returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 3840.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ImageGenerationDestinationConfig

Service: Amazon Kinesis Video Streams

The structure that contains the information required to deliver images to a customer.

Contents

DestinationRegion

The AWS Region of the S3 bucket where images will be delivered. This `DestinationRegion` must match the Region where the stream is located.

Type: String

Length Constraints: Minimum length of 9. Maximum length of 14.

Pattern: `^[a-z]+(-[a-z]+)?-[a-z]+-[0-9]$`

Required: Yes

Uri

The Uniform Resource Identifier (URI) that identifies where the images will be delivered.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `^[a-zA-Z_0-9]+:(//)?([^/]+)/?([^*]*)$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LastRecorderStatus

Service: Amazon Kinesis Video Streams

The latest status of a stream's edge recording job.

Contents

JobStatusDetails

A description of a recorder job’s latest status.

Type: String

Required: No

LastCollectedTime

The timestamp at which the recorder job was last executed and media stored to local disk.

Type: Timestamp

Required: No

LastUpdatedTime

The timestamp at which the recorder status was last updated.

Type: Timestamp

Required: No

RecorderStatus

The status of the latest recorder job.

Type: String

Valid Values: `SUCCESS | USER_ERROR | SYSTEM_ERROR`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LastUploaderStatus

Service: Amazon Kinesis Video Streams

The latest status of a stream's edge to cloud uploader job.

Contents

JobStatusDetails

A description of an uploader job's latest status.

Type: String

Required: No

LastCollectedTime

The timestamp at which the uploader job was last executed and media collected to the cloud.

Type: Timestamp

Required: No

LastUpdatedTime

The timestamp at which the uploader status was last updated.

Type: Timestamp

Required: No

UploaderStatus

The status of the latest uploader job.

Type: String

Valid Values: `SUCCESS | USER_ERROR | SYSTEM_ERROR`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ListEdgeAgentConfigurationsEdgeConfig

Service: Amazon Kinesis Video Streams

A description of a single stream's edge configuration.

Contents

CreationTime

The timestamp when the stream first created the edge config.

Type: Timestamp

Required: No

EdgeConfig

A description of the stream's edge configuration that will be used to sync with the Edge Agent IoT Greengrass component. The Edge Agent component will run on an IoT Hub Device setup at your premise.

Type: EdgeConfig object

Required: No

FailedStatusDetails

A description of the generated failure status.

Type: String

Required: No

LastUpdatedTime

The timestamp when the stream last updated the edge config.

Type: Timestamp

Required: No

StreamARN

The Amazon Resource Name (ARN) of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

SyncStatus

The current sync status of the stream's edge configuration.

Type: String

Valid Values: `SYNCING | ACKNOWLEDGED | IN_SYNC | SYNC_FAILED | DELETING | DELETE_FAILED | DELETING_ACKNOWLEDGED`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LocalSizeConfig

Service: Amazon Kinesis Video Streams

The configuration details that include the maximum size of the media (`MaxLocalMediaSizeInMB`) that you want to store for a stream on the Edge Agent, as well as the strategy that should be used (`StrategyOnFullSize`) when a stream's maximum size has been reached.

Contents

MaxLocalMediaSizeInMB

The overall maximum size of the media that you want to store for a stream on the Edge Agent.

Type: Integer

Valid Range: Minimum value of 64. Maximum value of 2000000.

Required: No

StrategyOnFullSize

The strategy to perform when a stream's `MaxLocalMediaSizeInMB` limit is reached.

Type: String

Valid Values: `DELETE_OLDEST_MEDIA | DENY_NEW_MEDIA`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MappedResourceConfigurationListItem

Service: Amazon Kinesis Video Streams

A structure that encapsulates, or contains, the media storage configuration properties.

Contents

ARN

The Amazon Resource Name (ARN) of the Kinesis Video Stream resource, associated with the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

Type

The type of the associated resource for the kinesis video stream.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MediaSourceConfig

Service: Amazon Kinesis Video Streams

The configuration details that consist of the credentials required (`MediaUriSecretArn` and `MediaUriType`) to access the media files that are streamed to the camera.

Contents

MediaUriSecretArn

The AWS Secrets Manager ARN for the username and password of the camera, or a local media file location.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:[a-z\d-]+:secretsmanager:[a-z0-9-]+:[0-9]+:secret:[a-zA-Z0-9_.-]+`

Required: Yes

MediaUriType

The Uniform Resource Identifier (URI) type. The `FILE_URI` value can be used to stream local media files.

> **Note**
>
> Preview only supports the `RTSP_URI` media source URI format .

Type: String

Valid Values: `RTSP_URI | FILE_URI`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MediaStorageConfiguration

Service: Amazon Kinesis Video Streams

A structure that encapsulates, or contains, the media storage configuration properties.

- If `StorageStatus` is enabled, the data will be stored in the `StreamARN` provided. In order for WebRTC Ingestion to work, the stream must have data retention enabled.
- If `StorageStatus` is disabled, no data will be stored, and the `StreamARN` parameter will not be needed.

Contents

Status

The status of the media storage configuration.

Type: String

Valid Values: `ENABLED | DISABLED`

Required: Yes

StreamARN

The Amazon Resource Name (ARN) of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++]

- AWS SDK for Java V2
- AWS SDK for Ruby V3

NotificationConfiguration

Service: Amazon Kinesis Video Streams

Use this API to configure Amazon Simple Notification Service (Amazon SNS) notifications for when fragments become available in a stream. If this parameter is null, the configuration will be deleted from the stream.

See Notifications in Kinesis Video Streams for more information.

Contents

DestinationConfig

The destination information required to deliver a notification to a customer.

Type: NotificationDestinationConfig object

Required: Yes

Status

Indicates if a notification configuration is enabled or disabled.

Type: String

Valid Values: `ENABLED | DISABLED`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

NotificationDestinationConfig

Service: Amazon Kinesis Video Streams

The structure that contains the information required to deliver a notification to a customer.

Contents

Uri

The Uniform Resource Identifier (URI) that identifies where the images will be delivered.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `^[a-zA-Z_0-9]+:(//)?([^/]+)/?([^*]*)$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

RecorderConfig

Service: Amazon Kinesis Video Streams

The recorder configuration consists of the local `MediaSourceConfig` details that are used as credentials to accesss the local media files streamed on the camera.

Contents

MediaSourceConfig

The configuration details that consist of the credentials required (`MediaUriSecretArn` and `MediaUriType`) to access the media files streamed to the camera.

Type: MediaSourceConfig object

Required: Yes

ScheduleConfig

The configuration that consists of the `ScheduleExpression` and the `DurationInMinutes` details that specify the scheduling to record from a camera, or local media file, onto the Edge Agent. If the `ScheduleExpression` attribute is not provided, then the Edge Agent will always be set to recording mode.

Type: ScheduleConfig object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ResourceEndpointListItem

Service: Amazon Kinesis Video Streams

An object that describes the endpoint of the signaling channel returned by the `GetSignalingChannelEndpoint` API.

The media server endpoint will correspond to the `WEBRTC` Protocol.

Contents

Protocol

The protocol of the signaling channel returned by the `GetSignalingChannelEndpoint` API.

Type: String

Valid Values: `WSS | HTTPS | WEBRTC`

Required: No

ResourceEndpoint

The endpoint of the signaling channel returned by the `GetSignalingChannelEndpoint` API.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ScheduleConfig

Service: Amazon Kinesis Video Streams

This API enables you to specify the duration that the camera, or local media file, should record onto the Edge Agent. The `ScheduleConfig` consists of the `ScheduleExpression` and the `DurationInMinutes` attributes.

If the `ScheduleConfig` is not provided in the `RecorderConfig`, then the Edge Agent will always be set to recording mode.

If the `ScheduleConfig` is not provided in the `UploaderConfig`, then the Edge Agent will upload at regular intervals (every 1 hour).

Contents

DurationInSeconds

The total duration to record the media. If the `ScheduleExpression` attribute is provided, then the `DurationInSeconds` attribute should also be specified.

Type: Integer

Valid Range: Minimum value of 60. Maximum value of 3600.

Required: Yes

ScheduleExpression

The Quartz cron expression that takes care of scheduling jobs to record from the camera, or local media file, onto the Edge Agent. If the `ScheduleExpression` is not provided for the `RecorderConfig`, then the Edge Agent will always be set to recording mode.

For more information about Quartz, refer to the *Cron Trigger Tutorial* page to understand the valid expressions and its use.

Type: String

Length Constraints: Minimum length of 11. Maximum length of 100.

Pattern: `[^\n]{11,100}`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

SingleMasterChannelEndpointConfiguration

Service: Amazon Kinesis Video Streams

An object that contains the endpoint configuration for the `SINGLE_MASTER` channel type.

Contents

Protocols

This property is used to determine the nature of communication over this `SINGLE_MASTER` signaling channel. If `WSS` is specified, this API returns a websocket endpoint. If `HTTPS` is specified, this API returns an `HTTPS` endpoint.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Valid Values: `WSS | HTTPS | WEBRTC`

Required: No

Role

This property is used to determine messaging permissions in this `SINGLE_MASTER` signaling channel. If `MASTER` is specified, this API returns an endpoint that a client can use to receive offers from and send answers to any of the viewers on this signaling channel. If `VIEWER` is specified, this API returns an endpoint that a client can use only to send offers to another `MASTER` client on this signaling channel.

Type: String

Valid Values: `MASTER | VIEWER`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

SingleMasterConfiguration

Service: Amazon Kinesis Video Streams

A structure that contains the configuration for the SINGLE_MASTER channel type.

Contents

MessageTtlSeconds

The period of time (in seconds) a signaling channel retains undelivered messages before they are discarded. Use UpdateSignalingChannel to update this value.

Type: Integer

Valid Range: Minimum value of 5. Maximum value of 120.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

StreamInfo

Service: Amazon Kinesis Video Streams

An object describing a Kinesis video stream.

Contents

CreationTime

A time stamp that indicates when the stream was created.

Type: Timestamp

Required: No

DataRetentionInHours

How long the stream retains data, in hours.

Type: Integer

Valid Range: Minimum value of 0.

Required: No

DeviceName

The name of the device that is associated with the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

KmsKeyId

The ID of the AWS Key Management Service (AWS KMS) key that Kinesis Video Streams uses to encrypt data on the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `.+`

Required: No

MediaType

The `MediaType` of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[\w\-\.\+]+/[\w\-\.\+]+(,[\w\-\.\+]+/[\w\-\.\+]+)*`

Required: No

Status

The status of the stream.

Type: String

Valid Values: `CREATING | ACTIVE | UPDATING | DELETING`

Required: No

StreamARN

The Amazon Resource Name (ARN) of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `arn:[a-z\d-]+:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName

The name of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Version

The version of the stream.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `[a-zA-Z0-9]+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

StreamNameCondition

Service: Amazon Kinesis Video Streams

Specifies the condition that streams must satisfy to be returned when you list streams (see the `ListStreams` API). A condition has a comparison operation and a value. Currently, you can specify only the `BEGINS_WITH` operator, which finds streams whose names start with a given prefix.

Contents

ComparisonOperator

A comparison operator. Currently, you can specify only the `BEGINS_WITH` operator, which finds streams whose names start with a given prefix.

Type: String

Valid Values: `BEGINS_WITH`

Required: No

ComparisonValue

A value to compare.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++]
- [AWS SDK for Java V2]
- [AWS SDK for Ruby V3]

Tag

Service: Amazon Kinesis Video Streams

A key and value pair that is associated with the specified signaling channel.

Contents

Key

The key of the tag that is associated with the specified signaling channel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

Required: Yes

Value

The value of the tag that is associated with the specified signaling channel.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: `[\p{L}\p{Z}\p{N}_.:/=+\-@]*`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

UploaderConfig

Service: Amazon Kinesis Video Streams

The configuration that consists of the `ScheduleExpression` and the `DurationInMinutes` details that specify the scheduling to record from a camera, or local media file, onto the Edge Agent. If the `ScheduleConfig` is not provided in the `UploaderConfig`, then the Edge Agent will upload at regular intervals (every 1 hour).

Contents

ScheduleConfig

The configuration that consists of the `ScheduleExpression` and the `DurationInMinutes` details that specify the scheduling to record from a camera, or local media file, onto the Edge Agent. If the `ScheduleConfig` is not provided in this `UploaderConfig`, then the Edge Agent will upload at regular intervals (every 1 hour).

Type: ScheduleConfig object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Amazon Kinesis Video Streams Media

The following data types are supported by Amazon Kinesis Video Streams Media:

- StartSelector

StartSelector

Service: Amazon Kinesis Video Streams Media

Identifies the chunk on the Kinesis video stream where you want the `GetMedia` API to start returning media data. You have the following options to identify the starting chunk:

- Choose the latest (or oldest) chunk.
- Identify a specific chunk. You can identify a specific chunk either by providing a fragment number or timestamp (server or producer).
- Each chunk's metadata includes a continuation token as a Matroska (MKV) tag (`AWS_KINESISVIDEO_CONTINUATION_TOKEN`). If your previous `GetMedia` request terminated, you can use this tag value in your next `GetMedia` request. The API then starts returning chunks starting where the last API ended.

Contents

StartSelectorType

Identifies the fragment on the Kinesis video stream where you want to start getting the data from.

- NOW - Start with the latest chunk on the stream.
- EARLIEST - Start with earliest available chunk on the stream.
- FRAGMENT_NUMBER - Start with the chunk after a specific fragment. You must also specify the `AfterFragmentNumber` parameter.
- PRODUCER_TIMESTAMP or SERVER_TIMESTAMP - Start with the chunk containing a fragment with the specified producer or server timestamp. You specify the timestamp by adding `StartTimestamp`.
- CONTINUATION_TOKEN - Read using the specified continuation token.

> **Note**
>
> If you choose the NOW, EARLIEST, or CONTINUATION_TOKEN as the `startSelectorType`, you don't provide any additional information in the `startSelector`.

Type: String

Valid Values: FRAGMENT_NUMBER | SERVER_TIMESTAMP | PRODUCER_TIMESTAMP | NOW | EARLIEST | CONTINUATION_TOKEN

Required: Yes

AfterFragmentNumber

Specifies the fragment number from where you want the GetMedia API to start returning the fragments.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: ^[0-9]+$

Required: No

ContinuationToken

Continuation token that Kinesis Video Streams returned in the previous GetMedia response. The GetMedia API then starts with the chunk identified by the continuation token.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: ^[a-zA-Z0-9_\.\-]+$

Required: No

StartTimestamp

A timestamp value. This value is required if you choose the PRODUCER_TIMESTAMP or the SERVER_TIMESTAMP as the startSelectorType. The GetMedia API then starts with the chunk containing the fragment that has the specified timestamp.

Type: Timestamp

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Amazon Kinesis Video Streams Archived Media

The following data types are supported by Amazon Kinesis Video Streams Archived Media:

- ClipFragmentSelector
- ClipTimestampRange
- DASHFragmentSelector
- DASHTimestampRange
- Fragment
- FragmentSelector
- HLSFragmentSelector
- HLSTimestampRange
- Image
- TimestampRange

ClipFragmentSelector

Service: Amazon Kinesis Video Streams Archived Media

Describes the timestamp range and timestamp origin of a range of fragments.

Fragments that have duplicate producer timestamps are deduplicated. This means that if producers are producing a stream of fragments with producer timestamps that are approximately equal to the true clock time, the clip will contain all of the fragments within the requested timestamp range. If some fragments are ingested within the same time range and very different points in time, only the oldest ingested collection of fragments are returned.

Contents

FragmentSelectorType

The origin of the timestamps to use (Server or Producer).

Type: String

Valid Values: `PRODUCER_TIMESTAMP | SERVER_TIMESTAMP`

Required: Yes

TimestampRange

The range of timestamps to return.

Type: ClipTimestampRange object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ClipTimestampRange

Service: Amazon Kinesis Video Streams Archived Media

The range of timestamps for which to return fragments.

Contents

EndTimestamp

The end of the timestamp range for the requested media.

This value must be within 24 hours of the specified `StartTimestamp`, and it must be later than the `StartTimestamp` value. If `FragmentSelectorType` for the request is `SERVER_TIMESTAMP`, this value must be in the past.

This value is inclusive. The `EndTimestamp` is compared to the (starting) timestamp of the fragment. Fragments that start before the `EndTimestamp` value and continue past it are included in the session.

Type: Timestamp

Required: Yes

StartTimestamp

The starting timestamp in the range of timestamps for which to return fragments.

Only fragments that start exactly at or after `StartTimestamp` are included in the session. Fragments that start before `StartTimestamp` and continue past it aren't included in the session. If `FragmentSelectorType` is `SERVER_TIMESTAMP`, the `StartTimestamp` must be later than the stream head.

Type: Timestamp

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++]()

- AWS SDK for Java V2
- AWS SDK for Ruby V3

DASHFragmentSelector

Service: Amazon Kinesis Video Streams Archived Media

Contains the range of timestamps for the requested media, and the source of the timestamps.

Contents

FragmentSelectorType

The source of the timestamps for the requested media.

When `FragmentSelectorType` is set to `PRODUCER_TIMESTAMP` and [GetDASHStreamingSessionURL:PlaybackMode](#) is `ON_DEMAND` or `LIVE_REPLAY`, the first fragment ingested with a producer timestamp within the specified [FragmentSelector:TimestampRange](#) is included in the media playlist. In addition, the fragments with producer timestamps within the `TimestampRange` ingested immediately following the first fragment (up to the [GetDASHStreamingSessionURL:MaxManifestFragmentResults](#) value) are included.

Fragments that have duplicate producer timestamps are deduplicated. This means that if producers are producing a stream of fragments with producer timestamps that are approximately equal to the true clock time, the MPEG-DASH manifest will contain all of the fragments within the requested timestamp range. If some fragments are ingested within the same time range and very different points in time, only the oldest ingested collection of fragments are returned.

When `FragmentSelectorType` is set to `PRODUCER_TIMESTAMP` and [GetDASHStreamingSessionURL:PlaybackMode](#) is `LIVE`, the producer timestamps are used in the MP4 fragments and for deduplication. But the most recently ingested fragments based on server timestamps are included in the MPEG-DASH manifest. This means that even if fragments ingested in the past have producer timestamps with values now, they are not included in the HLS media playlist.

The default is `SERVER_TIMESTAMP`.

Type: String

Valid Values: `PRODUCER_TIMESTAMP | SERVER_TIMESTAMP`

Required: No

TimestampRange

The start and end of the timestamp range for the requested media.

This value should not be present if `PlaybackType` is `LIVE`.

Type: DASHTimestampRange object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DASHTimestampRange

Service: Amazon Kinesis Video Streams Archived Media

The start and end of the timestamp range for the requested media.

This value should not be present if `PlaybackType` is `LIVE`.

The values in `DASHimestampRange` are inclusive. Fragments that start exactly at or after the start time are included in the session. Fragments that start before the start time and continue past it are not included in the session.

Contents

EndTimestamp

The end of the timestamp range for the requested media. This value must be within 24 hours of the specified `StartTimestamp`, and it must be later than the `StartTimestamp` value.

If `FragmentSelectorType` for the request is `SERVER_TIMESTAMP`, this value must be in the past.

The `EndTimestamp` value is required for `ON_DEMAND` mode, but optional for `LIVE_REPLAY` mode. If the `EndTimestamp` is not set for `LIVE_REPLAY` mode then the session will continue to include newly ingested fragments until the session expires.

> **Note**
>
> This value is inclusive. The `EndTimestamp` is compared to the (starting) timestamp of the fragment. Fragments that start before the `EndTimestamp` value and continue past it are included in the session.

Type: Timestamp

Required: No

StartTimestamp

The start of the timestamp range for the requested media.

If the `DASHTimestampRange` value is specified, the `StartTimestamp` value is required.

Only fragments that start exactly at or after `StartTimestamp` are included in the session. Fragments that start before `StartTimestamp` and continue past it aren't included in the session. If `FragmentSelectorType` is `SERVER_TIMESTAMP`, the `StartTimestamp` must be later than the stream head.

Type: Timestamp

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Fragment

Service: Amazon Kinesis Video Streams Archived Media

Represents a segment of video or other time-delimited data.

Contents

FragmentLengthInMilliseconds

The playback duration or other time value associated with the fragment.

Type: Long

Required: No

FragmentNumber

The unique identifier of the fragment. This value monotonically increases based on the ingestion order.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^[0-9]+$`

Required: No

FragmentSizeInBytes

The total fragment size, including information about the fragment and contained media data.

Type: Long

Required: No

ProducerTimestamp

The timestamp from the producer corresponding to the fragment, in milliseconds.

Type: Timestamp

Required: No

ServerTimestamp

The timestamp from the AWS server corresponding to the fragment, in milliseconds.

Type: Timestamp

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

FragmentSelector

Service: Amazon Kinesis Video Streams Archived Media

Describes the timestamp range and timestamp origin of a range of fragments.

Only fragments with a start timestamp greater than or equal to the given start time and less than or equal to the end time are returned. For example, if a stream contains fragments with the following start timestamps:

- 00:00:00
- 00:00:02
- 00:00:04
- 00:00:06

A fragment selector range with a start time of 00:00:01 and end time of 00:00:04 would return the fragments with start times of 00:00:02 and 00:00:04.

Contents

FragmentSelectorType

The origin of the timestamps to use (Server or Producer).

Type: String

Valid Values: `PRODUCER_TIMESTAMP | SERVER_TIMESTAMP`

Required: Yes

TimestampRange

The range of timestamps to return.

Type: TimestampRange object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

HLSFragmentSelector

Service: Amazon Kinesis Video Streams Archived Media

Contains the range of timestamps for the requested media, and the source of the timestamps.

Contents

FragmentSelectorType

The source of the timestamps for the requested media.

When `FragmentSelectorType` is set to `PRODUCER_TIMESTAMP` and GetHLSStreamingSessionURL:PlaybackMode is `ON_DEMAND` or `LIVE_REPLAY`, the first fragment ingested with a producer timestamp within the specified FragmentSelector:TimestampRange is included in the media playlist. In addition, the fragments with producer timestamps within the `TimestampRange` ingested immediately following the first fragment (up to the GetHLSStreamingSessionURL:MaxMediaPlaylistFragmentResults value) are included.

Fragments that have duplicate producer timestamps are deduplicated. This means that if producers are producing a stream of fragments with producer timestamps that are approximately equal to the true clock time, the HLS media playlists will contain all of the fragments within the requested timestamp range. If some fragments are ingested within the same time range and very different points in time, only the oldest ingested collection of fragments are returned.

When `FragmentSelectorType` is set to `PRODUCER_TIMESTAMP` and GetHLSStreamingSessionURL:PlaybackMode is `LIVE`, the producer timestamps are used in the MP4 fragments and for deduplication. But the most recently ingested fragments based on server timestamps are included in the HLS media playlist. This means that even if fragments ingested in the past have producer timestamps with values now, they are not included in the HLS media playlist.

The default is `SERVER_TIMESTAMP`.

Type: String

Valid Values: `PRODUCER_TIMESTAMP | SERVER_TIMESTAMP`

Required: No

TimestampRange

The start and end of the timestamp range for the requested media.

This value should not be present if `PlaybackType` is `LIVE`.

Type: HLSTimestampRange object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

HLSTimestampRange

Service: Amazon Kinesis Video Streams Archived Media

The start and end of the timestamp range for the requested media.

This value should not be present if `PlaybackType` is `LIVE`.

Contents

EndTimestamp

The end of the timestamp range for the requested media. This value must be within 24 hours of the specified `StartTimestamp`, and it must be later than the `StartTimestamp` value.

If `FragmentSelectorType` for the request is `SERVER_TIMESTAMP`, this value must be in the past.

The `EndTimestamp` value is required for `ON_DEMAND` mode, but optional for `LIVE_REPLAY` mode. If the `EndTimestamp` is not set for `LIVE_REPLAY` mode then the session will continue to include newly ingested fragments until the session expires.

> **Note**
>
> This value is inclusive. The `EndTimestamp` is compared to the (starting) timestamp of the fragment. Fragments that start before the `EndTimestamp` value and continue past it are included in the session.

Type: Timestamp

Required: No

StartTimestamp

The start of the timestamp range for the requested media.

If the `HLSTimestampRange` value is specified, the `StartTimestamp` value is required.

Only fragments that start exactly at or after `StartTimestamp` are included in the session. Fragments that start before `StartTimestamp` and continue past it aren't included in the session. If `FragmentSelectorType` is `SERVER_TIMESTAMP`, the `StartTimestamp` must be later than the stream head.

Type: Timestamp

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Image

Service: Amazon Kinesis Video Streams Archived Media

A structure that contains the `Timestamp`, `Error`, and `ImageContent`.

Contents

Error

The error message shown when the image for the provided timestamp was not extracted due to a non-tryable error. An error will be returned if:

- There is no media that exists for the specified `Timestamp`.
- The media for the specified time does not allow an image to be extracted. In this case the media is audio only, or the incorrect media has been ingested.

Type: String

Valid Values: `NO_MEDIA | MEDIA_ERROR`

Required: No

ImageContent

An attribute of the `Image` object that is Base64 encoded.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6291456.

Required: No

TimeStamp

An attribute of the `Image` object that is used to extract an image from the video stream. This field is used to manage gaps on images or to better understand the pagination window.

Type: Timestamp

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++]
- [AWS SDK for Java V2]
- [AWS SDK for Ruby V3]

TimestampRange

Service: Amazon Kinesis Video Streams Archived Media

The range of timestamps for which to return fragments.

Contents

EndTimestamp

The ending timestamp in the range of timestamps for which to return fragments.

Type: Timestamp

Required: Yes

StartTimestamp

The starting timestamp in the range of timestamps for which to return fragments.

Type: Timestamp

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Amazon Kinesis Video Signaling Channels

The following data types are supported by Amazon Kinesis Video Signaling Channels:

- IceServer

IceServer

Service: Amazon Kinesis Video Signaling Channels

A structure for the ICE server connection data.

Contents

Password

A password to login to the ICE server.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

Ttl

The period of time, in seconds, during which the user name and password are valid.

Type: Integer

Valid Range: Minimum value of 30. Maximum value of 86400.

Required: No

Uris

An array of URIs, in the form specified in the I-D.petithuguenin-behave-turn-uris spec. These URIs provide the different addresses and/or protocols that can be used to reach the TURN server.

Type: Array of strings

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

Username

A user name to login to the ICE server.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.-]+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Amazon Kinesis Video WebRTC Storage

The following data types are supported by Amazon Kinesis Video WebRTC Storage:

Common Errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

IncompleteSignature

The request signature does not conform to AWS standards.

HTTP Status Code: 400

InternalFailure

The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

InvalidAction

The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

InvalidClientTokenId

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

NotAuthorized

You do not have permission to perform this action.

HTTP Status Code: 400

OptInRequired

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

RequestExpired

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

ValidationError

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400

Common Parameters

The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more information about Signature Version 4, see Signing AWS API requests in the *IAM User Guide*.

Action

The action to be performed.

Type: string

Required: Yes

Version

The API version that the request is written for, expressed in the format YYYY-MM-DD.

Type: string

Required: Yes

X-Amz-Algorithm

The hash algorithm that you used to create the request signature.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Valid Values: `AWS4-HMAC-SHA256`

Required: Conditional

X-Amz-Credential

The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4_request"). The value is expressed in the following format: *access_key*/*YYYYMMDD*/*region*/*service*/aws4_request.

For more information, see Create a signed AWS API request in the *IAM User Guide*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-Date

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: `20120325T120000Z`.

Condition: X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see Elements of an AWS API request signature in the *IAM User Guide*.

Type: string

Required: Conditional

X-Amz-Security-Token

The temporary security token that was obtained through a call to AWS Security Token Service (AWS STS). For a list of services that support temporary security credentials from AWS STS, see AWS services that work with IAM in the *IAM User Guide*.

Condition: If you're using temporary security credentials from AWS STS, you must include the security token.

Type: string

Required: Conditional

X-Amz-Signature

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-SignedHeaders

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see [Create a signed AWS API request] in the *IAM User Guide*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional