# Table of Contents

What Is Amazon Kinesis Video Streams? ................................................................. 1  
Are You a First-Time User of Kinesis Video Streams? ............................................ 2  
How It Works ........................................................................................................... 3  
Producer Libraries .................................................................................................. 6  
API and Producer Libraries ....................................................................................... 4  
Kinesis Video Streams API ....................................................................................... 4  
Producer Libraries .................................................................................................. 6  
Controlling Access ................................................................................................... 7  
Policy Syntax ............................................................................................................. 7  
Actions for Kinesis Video Streams ......................................................................... 8  
Amazon Resource Names (ARNs) for Kinesis Video Streams ................................. 8  
Granting Other IAM Accounts Access to a Kinesis Video Stream .......................... 8  
Example Policies ..................................................................................................... 9  
Using Server-Side Encryption .................................................................................... 10  
What Is Server-Side Encryption for Kinesis Video Streams? ................................. 10  
Costs, Regions, and Performance Considerations ................................................... 11  
How Do I Get Started with Server-Side Encryption? ............................................. 11  
Creating and Using User-Generated AWS KMS Master Keys ............................... 12  
Permissions to Use User-Generated AWS KMS Master Keys ............................... 12  
Data Model ............................................................................................................... 13  
Stream Header Elements ......................................................................................... 14  
Frame Header Elements ........................................................................................... 17  
MKV Frame Data ...................................................................................................... 17  
Getting Started ........................................................................................................ 18  
Step 1: Set Up an Account ....................................................................................... 18  
Sign Up for AWS ....................................................................................................... 18  
Create an Administrator IAM User ......................................................................... 19  
Next Step .................................................................................................................. 19  
Step 2: Create a Kinesis Video Stream ................................................................... 19  
Create a Video Stream Using the Console ............................................................... 19  
Create a Video Stream Using the AWS CLI ......................................................... 21  
Next Step .................................................................................................................. 21  
Step 3: Run and Verify the Code ........................................................................... 21  
What's Next? ............................................................................................................ 21  
Producer Libraries .................................................................................................. 22  
Kinesis Video Streams Producer Client ................................................................. 22  
Kinesis Video Streams Producer Library ............................................................... 23  
Related Topics ......................................................................................................... 23  
Java Producer Library ............................................................................................... 23  
Procedure: Using the Java Producer SDK ............................................................... 24  
Step 1: Download and Configure the Code .............................................................. 24  
Step 2: Write and Examine the Code ..................................................................... 25  
Step 3: Run and Verify the Code ........................................................................... 26  
Android Producer Library ....................................................................................... 27  
Procedure: Using the Android Producer SDK ....................................................... 27  
Step 1: Download and Configure the Code .............................................................. 28  
Step 2: Examine the Code ...................................................................................... 29  
Step 3: Run and Verify the Code ........................................................................... 31  
C++ Producer Library .............................................................................................. 31  
Object Model .......................................................................................................... 32  
Putting Media into the Stream .............................................................................. 32  
Callback Interfaces ................................................................................................. 32  
Procedure: Using the C++ Producer SDK ............................................................... 32  
Step 1: Download and Configure the Code .............................................................. 34  
Step 2: Write and Examine the Code ................................................................... 35  
Step 3: Run and Verify the Code ........................................................................... 39
Error: "Illegal Instruction" when running GStreamer demo on Raspberry Pi ......................... 103
Camera fails to load on Raspberry Pi .................................................................................. 103
Camera can't be found on macOS High Sierra ................................................................. 104
jni.h file not found when compiling on macOS High Sierra ............................................. 104
Curl errors when running the GStreamer demo app ......................................................... 104
Time stamp/range assertion at run time on Raspberry Pi .................................................. 104
Assertion on gst_value_set_fraction_range_full on Raspberry Pi .................................... 104
Troubleshooting Stream Parser Library Issues ................................................................. 105
  Cannot access a single frame from the stream .............................................................. 105
  Fragment decoding error .............................................................................................. 105
Document History .............................................................................................................. 106
API Reference ................................................................................................................... 108
Actions ............................................................................................................................. 108
  Amazon Kinesis Video Streams .................................................................................. 108
  Amazon Kinesis Video Streams Media ................................................................. 137
  Amazon Kinesis Video Streams Archived Media ....................................................... 147
Data Types ....................................................................................................................... 153
  Amazon Kinesis Video Streams .................................................................................. 153
  Amazon Kinesis Video Streams Media ....................................................................... 157
  Amazon Kinesis Video Streams Archived Media ....................................................... 159
Common Errors ............................................................................................................... 162
Common Parameters ......................................................................................................... 164
What Is Amazon Kinesis Video Streams?

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

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, RADAR data, and more. As live video streams from these sources into a Kinesis video stream, you can build applications that can access the data, frame-by-frame, in real time for low-latency processing.

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 time stamps and ingestion time stamps. You can build applications that periodically batch-process the video data, or you can create applications that require ad hoc 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** – Kinesis Video Streams enables you to connect and stream video, audio, and other data from millions of devices ranging from consumer smartphones, drones, dash cams, and more. 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 time stamps. Your applications can easily retrieve specified data in a stream using the time-index.

- **Focus on managing applications instead of infrastructure** – Kinesis Video Streams is serverless, so there is no infrastructure to set up or manage. You don't need to worry about the deployment, configuration, or elastic scaling of the underlying infrastructure as 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 ad hoc 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 via the AWS Marketplace to process and analyze your streams. Kinesis Video Streams GET APIs enable you 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).
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. Amazon Kinesis Video Streams: How It Works (p. 3) – To learn about Kinesis Video Streams concepts.
2. Getting Started with Kinesis Video Streams (p. 18) – To set up your account and test Kinesis Video Streams.
3. Kinesis Video Streams Producer Libraries (p. 22) – To learn about creating a Kinesis Video Streams producer application.
4. Kinesis Video Stream Parser Library (p. 79) – To learn about processing incoming data frames in a Kinesis Video Streams consumer application.
5. Amazon Kinesis Video Streams Examples (p. 86) – To see more examples of what you can do with Kinesis Video Streams.
Amazon Kinesis Video Streams: How It Works

Topics

- Kinesis Video Streams API and Producer Libraries Support (p. 4)
- Controlling Access to Kinesis Video Streams Resources Using IAM (p. 7)
- Using Server-Side Encryption with Kinesis Video Streams (p. 10)
- Kinesis Video Streams Data Model (p. 13)

Amazon Kinesis Video Streams is a fully managed AWS service that enables you 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 smartphone 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 easy-to-use software and libraries that you can install and configure on your devices. These libraries make it easy 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 enables you to transport live video data, optionally store it, and make the data available for consumption both in real time and on a batch or ad hoc 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 durably stored and time-indexed when low latency processing is not required. You can create these consumer applications to run on Amazon EC2 instances.

- **Kinesis Video Stream Parser Library (p. 79)** – Enables Kinesis Video Streams applications to reliably get media from Kinesis video streams 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 (p. 4)**
- **Producer Libraries (p. 6)**

### 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 (p. 142)`.
As fragments arrive, Kinesis Video Streams assigns a unique fragment number, in increasing order. It also stores producer-side and server-side time stamps for each fragment, as Kinesis Video Streams-specific metadata.

- **Consumer APIs** – The following APIs enable consumers 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 (p. 138).

  **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 needs to return newer fragments that are not 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/server-side time stamps, 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 sub-lists 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 time stamps

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.

**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](#).
Controlling Access to Kinesis Video Streams Resources Using IAM

By using AWS Identity and Access Management (IAM) with Amazon Kinesis Video Streams, you can 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 (p. 7)
- Actions for Kinesis Video Streams (p. 8)
- Amazon Resource Names (ARNs) for Kinesis Video Streams (p. 8)
- Granting Other IAM Accounts Access to a Kinesis Video Stream (p. 8)
- Example Policies for Kinesis Video Streams (p. 9)

Policy Syntax

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

```json
{
   "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, IAM 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 need to 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, you might want to 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:


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 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 your stream (for example, 111111111111).
2. Create a managed policy on the account that owns the stream that allows the level of access that you want to grant. For example policies for Kinesis Video Streams resources, see Example Policies (p. 9) in the next section.
3. Create a role, specifying the account to which you are granting permissions, and attach the policy that you created in the previous step.
4. Create a managed policy that allows the `AssumeRole` action on the role you created in the previous step. For example, the role might look like the following:

```json
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::123456789012:role/CustomRole"
  }
}
```

For 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.

```json
{
  "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.

```json
{
  "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.

```json
{
   "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.

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

Using Server-Side Encryption with Kinesis Video Streams

Server-side encryption using AWS Key Management Service (AWS KMS) keys makes it easier for you 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? (p. 10)
- Costs, Regions, and Performance Considerations (p. 11)
- How Do I Get Started with Server-Side Encryption? (p. 11)
- Creating and Using User-Generated AWS KMS Master Keys (p. 12)
- Permissions to Use User-Generated AWS KMS Master Keys (p. 12)

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 at rest by using an AWS KMS customer master key (CMK) that you specify. Data is encrypted before it is
written to the Kinesis Video Streams stream storage layer, and it is decrypted after it is 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 master 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 master keys that are used by the server-side encryption feature. AWS KMS makes it easier to use a CMK for Kinesis Video Streams that is managed by AWS, a user-specified AWS KMS CMK, or a master 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 master keys, the (Default) aws/kinesis-video customer master key (CMK) is offered free of 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 CMK, including custom ones. The 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 CMK for Kinesis Video Streams that’s managed by AWS (alias = aws/kinesis-video) is free.
• User-generated AWS 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 is not specified when the stream is created, the default key (provided by Kinesis Video Streams) is used.

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

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

• When creating a Kinesis video stream in the AWS Management Console, specify the AWS KMS master key in the Encryption section on the Create new Kinesis Video stream page.
• When creating a Kinesis video stream using the CreateStream API, specify the key ID in the KmsKeyId parameter.
Creating and Using User-Generated AWS KMS Master Keys

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

Creating User-Generated AWS KMS Master Keys

For information about how to create your own master 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 KMS master key list.

Using User-Generated AWS KMS Master Keys

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

To use custom AWS KMS master 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 AWS KMS master keys from different accounts in streams created in the console.

Note
The AWS KMS key is not accessed until the PutMedia or GetMedia operation is executed. This has the following results:

• If the key you specify doesn’t exist, the CreateStream operation succeeds, but PutMedia and GetMedia operations on the stream fail.
• If you use the provided master key (aws/kinesis-video), the key is not present in your account until the first PutMedia or GetMedia operation is performed.

Permissions to Use User-Generated AWS KMS Master Keys

Before you can use server-side encryption with a user-generated AWS KMS master key, you must configure AWS 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 does not require application of custom IAM permissions.

Before you use user-generated AWS KMS master keys, ensure that your Kinesis video stream producers and consumers (IAM principals) are users in the AWS KMS master 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 AWS 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:

```json
{
    "Version": "2012-10-17",
    ...
```
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"
    }
  ]
}
```

Kinesis Video Streams Data Model

The Producer Libraries (p. 22) and Stream Parser Library (p. 79) 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 (p. 22) use 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 (p. 31).
Stream Header Elements

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

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Typical Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream_name</td>
<td>Corresponds to the name of the Kinesis video stream.</td>
<td>my-stream</td>
</tr>
<tr>
<td>retention_period</td>
<td>The duration that stream data is persisted by Kinesis Video Streams. Specify 0 for a stream that does not retain data.</td>
<td>24</td>
</tr>
<tr>
<td>tags</td>
<td>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 or get information about a stream.</td>
<td></td>
</tr>
<tr>
<td>kms_key_id</td>
<td>If present, the user-defined AWS KMS master key that is used to encrypt data on the stream. If it is absent, the data is encrypted by the Kinesis-supplied master key (aws/kinesis-video).</td>
<td>01234567-89ab-cdef-0123-456789ab</td>
</tr>
<tr>
<td>streaming_type</td>
<td>Currently, the only valid streaming type is STREAMING_TYPE_REALTIME.</td>
<td>STREAMING_TYPE_REALTIME</td>
</tr>
<tr>
<td>content_type</td>
<td>The user-defined content type. For streaming video data to play in the console, the content type must be video/h264.</td>
<td>video/h264</td>
</tr>
<tr>
<td>max_latency</td>
<td>This value is not currently used and should be set to 0.</td>
<td>0</td>
</tr>
<tr>
<td>fragment_duration</td>
<td>The estimate of how long your fragments should be, which is used for optimization. The actual fragment duration is determined by the streaming data.</td>
<td>2</td>
</tr>
<tr>
<td>timecode_scale</td>
<td>Indicates the scale used by frame time stamps. The default is 1 millisecond. Specifying 0</td>
<td></td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
<td>Typical Values</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>also assigns the default value of 1 millisecond. This value can be between 100 nanoseconds and 1 second. For more information, see TimecodeScale in the Matroska documentation.</td>
<td></td>
</tr>
<tr>
<td>key_frame_fragmentation</td>
<td>If true, the stream starts a new cluster when a keyframe is received.</td>
<td>true</td>
</tr>
<tr>
<td>frame_timecodes</td>
<td>If true, Kinesis Video Streams stamps the frames when they are received. If false, Kinesis Video Streams uses the decode time of the received frames.</td>
<td>true</td>
</tr>
<tr>
<td>absolute_fragment_time</td>
<td>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.</td>
<td>true</td>
</tr>
<tr>
<td>fragment_acks</td>
<td>If true, acknowledgements (ACKs) are sent when Kinesis Video Streams receives the data. The ACKs can be received using the KinesisVideoStreamFragmentAck or KinesisVideoStreamParseFragmentAck callbacks.</td>
<td>true</td>
</tr>
<tr>
<td>restart_on_error</td>
<td>Indicates whether the stream should resume transmission after a stream error is raised.</td>
<td>true</td>
</tr>
<tr>
<td>nal_adaptation_flags</td>
<td>Indicates whether NAL (Network Abstraction Layer) adaptation or codec private data is present in the content. Valid flags include NAL_ADAPTATION Annunci NALS and NAL_ADAPTATION Annunci CPD NALS.</td>
<td>NAL_ADAPTATION Annunci NALS</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
<td>Typical Values</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>avg_bandwidth_bps</td>
<td>An estimate of the content bandwidth. 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.</td>
<td>5</td>
</tr>
<tr>
<td>buffer_duration</td>
<td>The duration that content is to be buffered on the producer. If there is 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.</td>
<td></td>
</tr>
<tr>
<td>replay_duration</td>
<td>The amount of time the video data stream is &quot;rewound&quot; in the case of connection loss. 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 eliminate redundant frames. This value should be less than the buffer duration; otherwise the buffer duration is used.</td>
<td></td>
</tr>
<tr>
<td>connection_staleness</td>
<td>The duration that a connection is maintained when no data is received.</td>
<td></td>
</tr>
<tr>
<td>codec_id</td>
<td>The codec used by the content. For more information, see CodecID in the Matroska specification.</td>
<td>V_MPEG2</td>
</tr>
<tr>
<td>track_name</td>
<td>The user-defined name of the track.</td>
<td>my_track</td>
</tr>
</tbody>
</table>
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 time stamp of when this frame was decoded. If previous frames depend on this frame for decoding, this time stamp might be earlier than that of earlier frames. This value is relative to the start of the fragment.
- **Presentation Timestamp**: The time stamp 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 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.

Other sections in this guide describe how to send data to the stream and view data on the stream.

If you are new to Amazon Kinesis Video Streams, we recommend that you read Amazon Kinesis Video Streams: How It Works (p. 3) first.

Topics

- Step 1: Set Up an AWS Account and Create an Administrator (p. 18)
- Step 2: Create a Kinesis Video Stream (p. 19)
- What's Next? (p. 21)

Step 1: Set Up an AWS Account and Create an Administrator

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

1. Sign Up for AWS (p. 18) (unless you already have an account)
2. Create an Administrator IAM User (p. 19)

Sign Up for AWS

If you already have an AWS account, you can skip this step.

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including Kinesis Video Streams. When you use Kinesis Video Streams, you are charged based on the amount of data ingested into, stored by, and consumed from the service. If you are a new AWS customer, you can get started with Kinesis Video Streams for free. For more information, see AWS Free Usage Tier.

To create an AWS account

1. Open https://aws.amazon.com/, and then choose Create an AWS Account.
   
   Note
   This might be unavailable in your browser if you previously signed into the AWS Management Console. In that case, choose Sign in to a different account, and then choose Create a new AWS account.

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.
Create an Administrator IAM User

When you sign up for AWS, you provide an email address and password that is associated with your AWS account. This is your AWS account root user. Its credentials provide complete access to all of your AWS resources.

Note
For security reasons, we recommend that you use the root user only to create an administrator, which is an IAM user with full permissions to your AWS account. You can then use this administrator to create other IAM users and roles with limited permissions. For more information, see IAM Best Practices and Creating an Admin User and Group in the IAM User Guide.

To create an administrator and sign into the console

1. Create an administrator in your AWS account. For instructions, see Creating Your First IAM User and Administrators Group in the IAM User Guide.
2. As an administrator, you can sign in to the console using a special URL. For more information, see How Users Sign in to Your Account in the IAM User Guide.

The administrator can create more users in the account. IAM users by default don't have any permissions. The administrator can create users and manage their permissions. For more information, see Creating Your First IAM User and Administrators Group.

For more information about IAM, see the following:
- AWS Identity and Access Management (IAM)
- Getting Started
- IAM User Guide

Next Step

Step 2: Create a Kinesis Video Stream (p. 19)

Step 2: 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" (p. 19)
- the section called “Create a Video Stream Using the AWS CLI” (p. 21)

Create a Video Stream Using the Console

1. Sign in to the AWS Management Console and open the Kinesis console at https://console.aws.amazon.com/kinesis.
2. On the Manage streams page, choose Create.
3. On the Create new KinesisVideo Stream page, type ExampleStream for the stream name. Leave the Use default settings check box selected.
4. Choose **Create stream**.

5. After Kinesis Video Streams creates the stream, review the details on the **ExampleStream** page.
Create a Video Stream Using the AWS CLI

1. Ensure 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 "MyKinesisVideoStream" --data-retention-in-hours "24"
   ```

Next Step

What's Next? (p. 21)

What's Next?

After you have a video stream, you can start sending data to it from a Java application. In your code, use Kinesis Video Streams options to configure your application to extract data from your media sources and upload to your stream. For more information, see Using the Java Producer Library (p. 23).
Kinesis Video Streams Producer Libraries

The Amazon Kinesis Video Streams Producer libraries are a set of easy-to-use libraries that are part of 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 video and other media data.

Data can be streamed media in the following ways:

- Streaming media data in real time
- Streaming media data after buffering it for a few seconds
- Streaming after-the-fact media uploads

After you create a Kinesis Video Streams stream, you can start sending data to the stream. You can use the SDK to create application code that extracts the video data (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 (p. 22)
- Kinesis Video Streams Producer Library (p. 23)

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. Furthermore, it 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 does not 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 (p. 23) and Using the Android Producer Library (p. 27).

### 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 might have Java on the device, but you want to reduce the amount of overhead in your code and limit it to the bare minimum level of abstraction, due to limitations such as memory and processing power.

Currently, the Kinesis Video Streams Producer Library is available for C++ applications. For more information, see Using the C++ Producer Library (p. 31).

### Related Topics

- Using the Java Producer Library (p. 23)
- Using the Android Producer Library (p. 27)
- Using the C++ Producer Library (p. 31)

### Using the Java Producer Library

Amazon Kinesis Video Streams provides the Java Producer Library, which you can use to write application code, with minimal configuration, to send media data from a device to a Kinesis video stream.

You must 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 AWS 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 (not currently available)

  **Note**
  Pre-built libraries for macOS, Ubuntu, and Raspbian are available in src/main/resources/lib. For other environments, compile the C++ Producer Library (p. 31).

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.

   ```bash
   $ 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:
3. The Java example code uses the current AWS credentials. To use a different credentials profile, locate the following code in DemoAppMain.java:

```java
final KinesisVideoClient kinesisVideoClient = KinesisVideoJavaClientFactory.createKinesisVideoClient(
    Regions.US_WEST_2,
    AuthHelper.getSystemPropertiesCredentialsProvider());
```

Change the code to the following:

```java
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” (p. 25)

Step 2: Write and Examine the Code

In this section of the Java Producer Library procedure, you write and examine the Java example code.

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. That is, 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.

```java
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:
Creating an Instance of MediaSource

To send bytes to your Kinesis video stream, you need to 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.

```java
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).

```java
kinesisVideoClient.registerMediaSource(STREAM_NAME, bytesMediaSource);
```

Starting the Media Source

Start the media source so that it can begin generating data and sending it to the client.

```java
bytesMediaSource.start();
```

Next Step

the section called “Step 3: Run and Verify the Code” (p. 26)

Step 3: 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.secretAccessKey={YourAwsSecretKey} -Djava.library.path={NativeLibraryPath}''
     ```
   - For temporary AWS credentials: 
     ```
     -Daws.accessKeyId={YourAwsAccessKey} -Daws.secretAccessKey={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

Amazon Kinesis Video Streams provides the Android Producer Library, which you can use to write application code, with minimal configuration, to send media data from an Android device to a Kinesis video stream.

You must 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:

- Download and Configure the Code
- Examine the Code
- Run and Verify the Code

Prerequisites

- We recommend Android Studio for examining, editing, and running the application code. We recommend at least version 3.0.0, released October 2017.
- In the sample code, you provide Amazon Cognito credentials. Follow these procedures to set up an Amazon Cognito user pool and identity pool:

  To set up a user pool

  1. Sign in to the Amazon Cognito console.
  2. Choose Manage your User Pools.
  3. Choose Create a user pool.
  4. Type a value for Pool name; for example, <username>_android_user_pool.
  5. Choose Review defaults.
  6. Choose Create pool.
  7. Copy and save the Pool Id value. You will need this value when you configure the example application.
  8. On the page for your pool, choose App clients.
  9. Choose Add an app client.
  10. Type a value for App client name; for example, <username>_android_app_client.
11. Choose **Create app client**.

12. Choose **Show Details**, and copy and save the **App client ID** and **App client secret**. You will need these values when you configure the example application.

**To set up an identity pool**

1. Open the Amazon Cognito console.
2. Choose **Manage Federated Identities**.
3. Choose **Create new identity pool**.
4. Type a value for **Identity pool name**; for example, `<username>_android_identity_pool`.
5. Expand the **Authentication providers** section. On the Cognito tab, add the values for the **User Pool ID** and **App client ID** from the previous procedure.
6. Choose **Create pool**.
7. On the next page, expand the **Show Details** section.
8. In the section that has a value for **Role name** that ends in **Auth_Role**, choose **View Policy Document**.
9. Choose **Edit**, and confirm the **Edit Policy** dialog box that appears. Then copy the following JSON and paste it into the editor:

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

10. Choose **Allow**.

11. On the next page, copy and save the **Identity pool ID** value from the **Get AWS Credentials** code snippet. You will need this value when you configure the example application.

**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 Android SDK from the GitHub repository.

   ```bash
   $ 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**.


   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:

   ```json
   {
     "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": "0123456789abcdefgihjklmno",
         "PoolId": "us-west-2_qRsTuVwXy",
         "Region": "us-west-2"
       }
     }
   }
   ```

7. Update the `StreamingFragment.java` file with your region:

   ```java
   try {
     mKinesisVideoClient = KinesisVideoAndroidClientFactory.createKinesisVideoClient(
       getActivity(),
       Regions.US_WEST_2,
       KinesisVideoDemoApp.getCredentialsProvider());
   }
   
   For AWS region constants, see [Regions](#).
   ```

**Next Step**

the section called “**Step 2: Examine the Code**” (p. 29)

**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—that is, 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.

```java
mKinesisVideoClient = KinesisVideoAndroidClientFactory.createKinesisVideoClient(
    getActivity(),
    Regions.US_WEST_2,
    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:

```java
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:

```java
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” (p. 31)

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 +, Android App. In the Name field, enter AmazonKinesisVideoDemoApp. In the Module pulldown, choose AmazonKinesisVideoDemoApp. Choose OK.
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 you receive by email, and choose Confirm. Choose Ok.
9. On the next page, leave the default values, and choose Stream.
10. Sign in to the AWS Management Console and open the Kinesis Video Streams console at https://console.aws.amazon.com/kinesisvideo/ 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

Amazon Kinesis Video Streams provides the C++ Producer Library, which you can use 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, media content type, and so on.

Putting Media into the Stream

The C++ library provides methods (for example, `PutFrame`) that you can use 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, deletes a stream, and so on.
- 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.

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.

**Note**

The C++ library includes a sample build script for macOS. The C++ Producer Library is not currently available for Windows.

To use the C++ Producer Library on a Raspberry Pi device, see [Appendix: Using the C++ Producer SDK on Raspberry Pi](p. 39).

The procedure includes the following steps:

- **Step 1: Download and Configure the Code**
Procedure: Using the C++ Producer SDK

- **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 certification 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**
  - **Flex 2.5.35 Apple (flex-31) or later**
  - **Bison 2.4** (GNU License)
  - **Automake 1.15.1** (GNU License)
  - **GNU Libtool (Apple Inc. version cctools-898)**
  - **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 (responses to version commands are truncated):

  - **Install Git:** `sudo apt-get install git`
    ```bash
    # git --version
    git version 2.14.1
    ```
  - **Install CMake:** `sudo apt-get install cmake`
    ```bash
    # cmake --version
    cmake version 3.9.1
    ```
  - **Install Libtool:** `sudo apt-get install libtool`
    ```bash
    2.4.6-2
    ```
  - **Install libtool-bin:** `sudo apt-get install libtool-bin`
    ```bash
    # libtool --version
    libtool (GNU libtool) 2.4.6
    Written by Gordon Matzigkeit, 1996
    ```
  - **Install GNU Automake:** `sudo apt-get install automake`
    ```bash
    # automake --version
    automake (GNU automake) 1.15
    ```
Step 1: Download and Configure the Code

- Install GNU Bison: `sudo apt-get install bison`
  
  ```
  $ bison -V
  bison (GNU Bison) 3.0.4
  ```

- Install G++: `sudo apt-get install g++`
  
  ```
  g++ --version
  g++ (Ubuntu 7.2.0-8ubuntu3) 7.2.0
  ```

- Install curl: `sudo apt-get install curl`
  
  ```
  $ curl --version
  curl 7.55.1 (x86_64-pc-linux-gnu) libcurl/7.55.1 OpenSSL/1.0.2g zlib/1.2.11
  libidn2/2.0.2 libpsl/0.18.0 (+libidn2/2.0.2) librtmp/2.3
  ```

- Install pkg-config: `sudo apt-get install pkg-config`
  
  ```
  $ pkg-config --version
  0.29.1
  ```

- Install Flex: `sudo apt-get install flex`
  
  ```
  $ flex --version
  flex 2.6.1
  ```

- Install OpenJDK: `sudo apt-get install openjdk-8-jdk`
  
  ```
  $ java -showversion
  openjdk version "1.8.0_151"
  ```

- Set the `JAVA_HOME` environment variable: `export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/`

- Run the build script: `./install-script`

Next Step

Step 1: Download and Configure the C++ Producer Library Code

In this section, you download the low-level libraries and configure the application to use your AWS credentials.

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

2. Open the code in the integrated development environment (IDE) of your choice (for example, Eclipse).
### 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).

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 details:

#### 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:

```cpp
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.

---

3. At the command line, set the `ACCESS_KEY_ENV_VAR` and `SECRET_KEY_ENV_VAR` environment variables to your AWS credentials. Alternatively, you can hardcode your AWS credentials in the following lines of `ProducerTestFixture.h`:

```cpp
if (nullptr == (accessKey = getenv(ACCESS_KEY_ENV_VAR))) {
    accessKey = "AccessKey";
}
if (nullptr == (secretKey = getenv(SECRET_KEY_ENV_VAR))) {
    secretKey = "SecretKey";
}
```

4. In `tst/ProducerTestFixture.h`, find the call to `CreateStream`. Change the name of the stream definition from `ScaryTestStream2` to a unique name:

```cpp
shared_ptr<KinesisVideoStream> CreateTestStream(int index) {
    char stream_name[MAX_STREAM_NAME_LEN];
    sprintf(stream_name, "ScaryTestStream_%d", index);
    return share Thiên
```
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, and then multiply by 1.2 to account for defragmentation. For example, if your application has the following configuration:

- Three streams
- 3 minutes of maximum duration
- Each stream is 30 fps
- Each frame is 10,000 KB in size

The content store requirement for this application is

\[
3 \text{ (streams)} \times 3 \text{ (minutes)} \times 60 \text{ (seconds in a minute)} \times 10000 \text{ (kb)} \times 1.2 \text{ (defragmentation allowance)} = 194.4 \text{ Mb} \approx 200\text{Mb}.
\]

- 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:

```cpp
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. To view the stream in the console viewer, set this value to "video/h264".
- Media latency. This value is not currently used, and should be set to 0.
- Playback duration of each fragment.
- Media timecode scale.
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:

```cpp
frame.duration = FRAME_DURATION_IN_MILLISECONDS * HUNDREDS_OF_NANOS_IN_A_MILLISECOND;
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 time stamp.**
- **Presentation time stamp.**
- **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](#).

**Metrics and Metric Logging**

The C++ Producer SDK includes functionality for metrics and metric logging.

You can use the `getKinesisVideoMetrics` and `getKinesisVideoStreamMetrics` APIs 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.

```c
/**
 * 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 free 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 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 being 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.

**Next Step**

the section called “Step 3: Run and Verify the Code” (p. 39)
Step 3: Run and Verify the Code

To run and verify the code for the C++ Producer Library procedure, do the following:

1. See Prerequisites for credential, certificate, and build requirements.
2. Build the project by using the /kinesis-video-native-build/install-script script. Running the install script installs the following open source dependencies:
   - curl lib
   - openssl (crypto and ssl)
   - log4cplus
   - jsoncpp

   **Note**
   To configure log4cplus, set the following value in PlatformUtils.h to point to your logging function:
   ```
   #define __LOG(p1, p2, p3, ...)     printf(p3, ##__VA_ARGS__)
   ```

3. The executable is built in kinesis-video-native-build/start. Launch it to run the unit test and kick off dummy frame streaming.
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**
   Because the test harness only sends frames of empty bytes, the console doesn't display the data as a video stream.

Appendix: 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 (p. 40)
- Create an IAM User with Permission to Write to Kinesis Video Streams (p. 40)
- Join Your Raspberry Pi to Your Wi-Fi Network (p. 41)
- Connect Remotely to Your Raspberry Pi (p. 41)
- Configure the Raspberry Pi Camera (p. 42)
- Install Software Prerequisites (p. 42)
- Download and Build the Kinesis Video Streams C++ Producer SDK (p. 43)
- Stream Video to Your Kinesis Video Stream and View the Live Stream (p. 43)
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 Raspbian image from the Raspberry Pi Foundation 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.

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.

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.
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.

```json
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "kinesisvideo:DescribeStream",
      "kinesisvideo:CreateStream",
      "kinesisvideo:GetDataEndpoint",
      "kinesisvideo:PutMedia"
    ],
    "Resource": ["*"
      ]
  }]
}
```
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 (p. 42).

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 (or download a sample wpa_supplicant.conf file):

   ```
country=US
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
  ssid="<YOUR_WIFI_SSID>"
  scan_ssid=1
  key_mgmt=WPA-PSK
  psk="<YOUR_WIFI_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 (p. 42).

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.
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:

   ```bash
   # sudo nano /etc/modules
   ```

2. Add the following line to the end of the file, if it's not already there:

   ```bash
   bcm2835-v4l2
   ```

3. Save the file and exit the editor (Ctrl-X).

4. Reboot the Raspberry Pi:

   ```bash
   # sudo reboot
   ```

5. When the device reboots, connect to it again through your terminal application if you are connecting remotely.

6. Open `raspi-config`:

   ```bash
   # sudo raspi-config
   ```

7. Choose 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:

   ```bash
   # raspistill -v -o test.jpg
   ```

   The display shows a five-second preview from the camera, takes a picture (saved to `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. Install Git:

   ```bash
   # sudo apt-get update
   # sudo apt-get install git
   ```

2. Install Yacc, Lex, and OpenJDK (Open Java Development Kit):
Using the C++ Producer SDK on Raspberry Pi

3. Set the `JAVA_HOME` environment variable:

   ```bash
   $ export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-armhf/
   ```

   **Note**
   If you reboot the device before building the SDK, you must repeat this step. You can also set
   this environment variable in your `~/.profile` file.

4. CMake is used to build the SDK. Install CMake with the following command:

   ```bash
   $ sudo apt-get install cmake
   ```

5. Copy the following PEM file to `/etc/ssl/cert.pem`:

   ```bash
   https://www.amazontrust.com/repository/SFSRootCA2.pem
   ```

   **Download and Build the Kinesis Video Streams C++ Producer SDK**

1. Install the C++ Producer SDK:

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

2. Change your current working directory to the install directory:

   ```bash
   $ cd amazon-kinesis-video-stream-producer-sdk-cpp/kinesis-video-native-build
   ```

3. Make the install script executable:

   ```bash
   $ chmod +x ./kinesis-video-native-build/install-script
   ```

4. Run the install script. The script downloads the source and builds several open-source projects. It
   might take several hours to run the first time it is executed:

   ```bash
   $ ./install-script
   ```

5. Type `Y` to verify. Then the build script runs.

   **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 (p. 40) 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 (p. 40).

2. Run the sample application using the following command:

   ```bash
   $ export AWS_ACCESS_KEY_ID=<Access Key ID>
   ```
3. You can specify the image size, framerate, and bitrate as follows:

```bash
$ export AWS_ACCESS_KEY_ID=<Access Key ID>
export AWS_SECRET_ACCESS_KEY=<Secret Access Key>
./kinesis_video_gstreamer_sample_app -w <width> -h <height> -f <framerate> -b <bitrateInKBPS> Stream Name
```

4. If the sample application exits with a `library not found` error, type the following commands to verify that the project is correctly linked to its open-source dependencies:

```bash
$ rm -rf ./kinesis-video-native-build/CMakeCache.txt ./kinesis-video-native-build/CMakeFiles
$ ./kinesis-video-native-build/install-script
```

5. Open the Kinesis Video Streams console at [https://console.aws.amazon.com/kinesisvideo/](https://console.aws.amazon.com/kinesisvideo/).

6. 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 bit rate 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 (p. 22).

### Topics

- Producer SDK Limits (p. 44)
- Error Code Reference (p. 46)
- Network Abstraction Layer (NAL) Adaptation Flag Reference (p. 65)
- Producer SDK Structures (p. 66)
- KVS Stream Structures (p. 67)

### Producer SDK Limits

The following table contains the current limits for values in the Producer Libraries (p. 22).
<table>
<thead>
<tr>
<th>Value</th>
<th>Limit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max stream count</td>
<td>128</td>
<td>The maximum number of streams that a producer object can create. This is a soft limit (you can request an increase). It ensures that the producer doesn't accidentally create streams recursively.</td>
</tr>
<tr>
<td>Max device name length</td>
<td>128 characters</td>
<td></td>
</tr>
<tr>
<td>Max tag count</td>
<td>50 per stream</td>
<td></td>
</tr>
<tr>
<td>Max stream name length</td>
<td>256 characters</td>
<td></td>
</tr>
<tr>
<td>Min storage size</td>
<td>10 MiB = 10 * 1024 * 1024 bytes</td>
<td></td>
</tr>
<tr>
<td>Max storage size</td>
<td>10 GiB = 10 * 1024 * 1024 * 1024 bytes</td>
<td></td>
</tr>
<tr>
<td>Max root directory path length</td>
<td>4,096 characters</td>
<td></td>
</tr>
<tr>
<td>Max auth info length</td>
<td>10,000 bytes</td>
<td></td>
</tr>
<tr>
<td>Max URI string length</td>
<td>10,000 characters</td>
<td></td>
</tr>
<tr>
<td>Max tag name length</td>
<td>128 characters</td>
<td></td>
</tr>
<tr>
<td>Max tag value length</td>
<td>1,024 characters</td>
<td></td>
</tr>
<tr>
<td>Min security token period</td>
<td>30 seconds</td>
<td></td>
</tr>
<tr>
<td>Security token grace period</td>
<td>40 minutes</td>
<td>If the specified duration is longer, it is limited to this value.</td>
</tr>
<tr>
<td>Retention period</td>
<td>0 or greater than one hour</td>
<td>0 indicates no retention.</td>
</tr>
<tr>
<td>Min cluster duration</td>
<td>1 second</td>
<td>The value is specified in 100 ns units, which is the SDK standard.</td>
</tr>
<tr>
<td>Max cluster duration</td>
<td>30 seconds</td>
<td>The value is specified in 100 ns units, which is the SDK standard. The backend API may enforce a shorter cluster duration.</td>
</tr>
<tr>
<td>Max fragment size</td>
<td>50 MB</td>
<td>For more information, see Kinesis Video Streams Limits (p. 97).</td>
</tr>
<tr>
<td>Max fragment duration</td>
<td>10 seconds</td>
<td>For more information, see Kinesis Video Streams Limits (p. 97).</td>
</tr>
<tr>
<td>Max connection duration</td>
<td>45 minutes</td>
<td>The backend closes the connection after this time. The SDK rotates the token and establishes a new connection within this time.</td>
</tr>
<tr>
<td>Value</td>
<td>Limit</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Max ACK segment length</td>
<td>1,024 characters</td>
<td>Maximum segment length of the acknowledgement sent to the ACK parser function.</td>
</tr>
<tr>
<td>Max content type string length</td>
<td>128 characters</td>
<td></td>
</tr>
<tr>
<td>Max codec ID string length</td>
<td>32 characters</td>
<td></td>
</tr>
<tr>
<td>Max track name string length</td>
<td>32 characters</td>
<td></td>
</tr>
<tr>
<td>Max codec private data length</td>
<td>1 MiB = 1 * 1024 * 1024 bytes</td>
<td></td>
</tr>
<tr>
<td>Min timecode scale value length</td>
<td>100 ns</td>
<td>The minimum timecode scale value to represent the frame time stamps in the resulting MKV cluster. The value is specified in increments of 100 ns, which is the SDK standard.</td>
</tr>
<tr>
<td>Max timecode scale value length</td>
<td>1 second</td>
<td>The maximum timecode scale value to represent the frame time stamps in the resulting MKV cluster. The value is specified in increments of 100 ns, which is the SDK standard.</td>
</tr>
<tr>
<td>Min content view item count</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Min buffer duration</td>
<td>20 seconds</td>
<td>The value is specified in increments of 100 ns, which is the SDK standard.</td>
</tr>
<tr>
<td>Max update version length</td>
<td>128 characters</td>
<td></td>
</tr>
<tr>
<td>Max ARN length</td>
<td>1024 characters</td>
<td></td>
</tr>
<tr>
<td>Max fragment sequence length</td>
<td>128 characters</td>
<td></td>
</tr>
<tr>
<td>Max retention period</td>
<td>10 years</td>
<td></td>
</tr>
</tbody>
</table>

**Error Code Reference**

This section contains error and status code information for the Producer Libraries (p. 22).

For information about solutions to common issues, see Troubleshooting Kinesis Video Streams (p. 99).

**Errors and Status Codes Returned by PutFrame Callbacks**

The following sections contain error and status information that is returned by callbacks for the PutFrame operation.

**Topics**

- Error and Status Codes Returned by the Client Library (p. 47)
- Error and Status Codes Returned by the Duration Library (p. 59)
- Error and Status Codes Returned by the Common Library (p. 59)
## Error and Status Codes Returned by the Client Library

The following table contains error and status information that is returned by methods in the Kinesis Video Streams Client library.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
<th>Recommended Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x52000001</td>
<td>STATUS_MAX_STREAM_COUNT</td>
<td>The maximum stream count was reached.</td>
<td>Specify a larger max stream count in DeviceInfo as specified in Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x52000002</td>
<td>STATUS_MIN_STREAM_COUNT</td>
<td>Minimum stream count error.</td>
<td>Specify the max number of streams greater than 0 in DeviceInfo.</td>
</tr>
<tr>
<td>0x52000003</td>
<td>STATUS_INVALID_DEVICE_NAME_LENGTH</td>
<td>Invalid device name length.</td>
<td>Refer to the max device name length in characters that is specified in Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x52000004</td>
<td>STATUS_INVALID_DEVICE_INFO_VERSION</td>
<td>Invalid DeviceInfo structure version.</td>
<td>Specify the correct current version of the structure.</td>
</tr>
<tr>
<td>0x52000005</td>
<td>STATUS_MAX_TAG_COUNT</td>
<td>The maximum tag count was reached.</td>
<td>Refer to the current max tag count that is specified in Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x52000006</td>
<td>STATUS_DEVICE_FINGERPRINT_LENGTH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x52000007</td>
<td>STATUS_INVALID_CALLBACKS_VERSION</td>
<td>Invalid Callbacks structure version.</td>
<td>Specify the correct current version of the structure.</td>
</tr>
<tr>
<td>0x52000008</td>
<td>STATUS_INVALID_STREAM_INFO_VERSION</td>
<td>Invalid StreamInfo structure version.</td>
<td>Specify the correct current version of the structure.</td>
</tr>
<tr>
<td>0x52000009</td>
<td>STATUS_INVALID_STREAM_NAME_LENGTH</td>
<td>Invalid stream name length.</td>
<td>Refer to the max stream name length in characters that is specified in Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x5200000a</td>
<td>STATUS_INVALID_STORAGE_SIZE</td>
<td>An invalid storage size was specified.</td>
<td>The storage size in bytes must be within the limits specified in Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x5200000b</td>
<td>STATUS_INVALID_ROOT_DIRECTORY_LENGTH</td>
<td>Invalid root directory string length.</td>
<td>Refer to the max root directory path length that is specified in Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
<td>Recommended Action</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x5200000c</td>
<td>STATUS_INVALID_SPILL_RATIO</td>
<td>Invalid spill ratio.</td>
<td>Express the spill ratio as a percentage from 0 to 100.</td>
</tr>
<tr>
<td>0x5200000d</td>
<td>STATUS_INVALID_STORAGE_INFO_STRUCTURE_VERSION</td>
<td>Invalid StorageInfo structure version.</td>
<td>Specify the correct current version of the structure.</td>
</tr>
<tr>
<td>0x5200000e</td>
<td>STATUS_INVALID_STREAM_STATE</td>
<td>The stream is in a state that doesn't permit the current operation.</td>
<td>Most commonly, this error occurs when the SDK fails to reach the state that it needs 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.</td>
</tr>
<tr>
<td>0x5200000f</td>
<td>STATUS_SERVICE_CALL_CALLBACKS_MISSING</td>
<td>The Callbacks structure has missing function entry points for some mandatory functions.</td>
<td>Ensure that the mandatory callbacks are implemented in the client application. This error is exposed only to PIC (Platform Independent Code) clients. C++ and other higher-level wrappers satisfy these calls.</td>
</tr>
<tr>
<td>0x52000010</td>
<td>STATUS_SERVICE_CALL_NOT_AUTHORIZED_ERROR</td>
<td>Not authorized.</td>
<td>Verify the security token/certificate/security token integration/expiration. Ensure that the token has the correct associated rights with it. For the Kinesis Video Streams sample applications, ensure that the environment variable is set correctly.</td>
</tr>
<tr>
<td>0x52000011</td>
<td>STATUS_DESCRIBE_STREAM_CALL_FAILED</td>
<td>DescribeStream API failure.</td>
<td>This error is returned after the DescribeStream API retry failure. The PIC client returns this error after it gives up retrying.</td>
</tr>
<tr>
<td>0x52000012</td>
<td>STATUS_INVALID_DESCRIBE_STREAM_RESPONSE</td>
<td>The structure that was passed to the DescribeStreamResultEvent structure.</td>
<td>The structure that was passed to the DescribeStreamResultEvent structure is either null or contains invalid items like a null Amazon Resource Name (ARN).</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
<td>Recommended Action</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------</td>
<td>--------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x52000013</td>
<td>STATUS_STREAM_IS_BEING_DELETED</td>
<td>The stream is being deleted.</td>
<td>An API failure was caused by the stream being deleted. Ensure that no other processes are trying to delete the stream while the stream is in use.</td>
</tr>
<tr>
<td>0x52000014</td>
<td>STATUS_SERVICE_CALL_INVALID_ARG</td>
<td>Invalid arguments were specified for the service call.</td>
<td>The backend returns this error when a service call argument is not valid or when the SDK encounters an error that it can't interpret.</td>
</tr>
<tr>
<td>0x52000015</td>
<td>STATUS_SERVICE_CALL_DEVICE_NOT_FOUND</td>
<td>The device was not found.</td>
<td>Ensure that the device is not deleted while in use.</td>
</tr>
<tr>
<td>0x52000016</td>
<td>STATUSSERVICE_CALL_DEVICE_NOT_PROVISIONED</td>
<td>The device was not provisioned.</td>
<td>Ensure that the device has been provisioned.</td>
</tr>
<tr>
<td>0x52000017</td>
<td>STATUS_SERVICE_CALL_RESOURCE_NOT_FOUND</td>
<td>Generic resource not found returned from the service.</td>
<td>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 ensures that the stream is created first.</td>
</tr>
<tr>
<td>0x52000018</td>
<td>STATUS_INVALID_AUTH_LEN</td>
<td>Invalid auth info length.</td>
<td>Refer to the current values that are specified in Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x52000019</td>
<td>STATUS_CREATE_STREAM_CALL_FAILED</td>
<td>The CreateStream API call failed.</td>
<td>Refer to the error string for more detailed information about why the operation failed.</td>
</tr>
<tr>
<td>0x5200002a</td>
<td>STATUS_GET_STREAMING_TOKEN_CALL_FAILED</td>
<td>GetStreamingToken call failed.</td>
<td>Refer to the error string for more detailed information about why the operation failed.</td>
</tr>
<tr>
<td>0x5200002b</td>
<td>STATUS_GET_STREAMING_ENDPOINT_CALL_FAILED</td>
<td>GetStreamingEndpoint API call failed.</td>
<td>Refer to the error string for more detailed information about why the operation failed.</td>
</tr>
<tr>
<td>0x5200002c</td>
<td>STATUS_INVALID_URI_LEN</td>
<td>An invalid URI string length was returned from the GetStreamingEndpoint API.</td>
<td>Refer to the current maximum values that are specified in Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
<td>Recommended Action</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x5200002d</td>
<td>STATUS_PUT_STREAM_CALL_FAILED</td>
<td>The PutMedia API call failed.</td>
<td>Refer to the error string for more detailed information about why the operation failed.</td>
</tr>
<tr>
<td>0x5200002e</td>
<td>STATUS_STORE_OUT_OF_MEMORY</td>
<td>The content store is out of memory.</td>
<td>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. It's better to drop 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 triggers 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.</td>
</tr>
<tr>
<td>0x5200002f</td>
<td>STATUS_NO_MORE_DATA_AVAILABLE</td>
<td>No more data is available currently for a stream.</td>
<td>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) do not see this warning because it's handled internally.</td>
</tr>
<tr>
<td>0x52000030</td>
<td>STATUS_INVALID_TAGVERSION</td>
<td>Invalid Tag structure version.</td>
<td>Specify the correct current version of the structure.</td>
</tr>
<tr>
<td>0x52000031</td>
<td>STATUS_SERVICECALL.UNKNOWN_ERROR</td>
<td>An unknown or generic error was returned from the networking stack.</td>
<td>See the logs for more detailed information.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
<td>Recommended Action</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------</td>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x52000032</td>
<td>STATUS_SERVICE_CALL_Resource_in_use_ERROR</td>
<td>Returned from the service. For more information, see the Kinesis Video Streams API Reference.</td>
<td></td>
</tr>
<tr>
<td>0x52000033</td>
<td>STATUS_SERVICE_CALL_Client_limit_ERROR</td>
<td>Returned from the service. For more information, see the Kinesis Video Streams API Reference.</td>
<td></td>
</tr>
<tr>
<td>0x52000034</td>
<td>STATUS_SERVICE_CALL_Device_limit_ERROR</td>
<td>Returned from the service. For more information, see the Kinesis Video Streams API Reference.</td>
<td></td>
</tr>
<tr>
<td>0x52000035</td>
<td>STATUS_SERVICE_CALL_Stream_limit_ERROR</td>
<td>Returned from the service. For more information, see the Kinesis Video Streams API Reference.</td>
<td></td>
</tr>
<tr>
<td>0x52000036</td>
<td>STATUS_SERVICE_CALL_Resource_deleted_ERROR</td>
<td>Returned from the service. For more information, see the Kinesis Video Streams API Reference.</td>
<td></td>
</tr>
<tr>
<td>0x52000037</td>
<td>STATUS_STREAM_READY_CALLBACK_FAILED</td>
<td>Calling a particular service API resulted in a timeout. Ensure that you have a valid network connection. The PIC will retry the operation automatically.</td>
<td></td>
</tr>
<tr>
<td>0x52000038</td>
<td>STATUS_DEVICE_TAGS_COUNT_NON_ZERO_TAGS_NULL</td>
<td>This notification is sent from the PIC to the client indicating that the async stream has been created.</td>
<td></td>
</tr>
<tr>
<td>0x52000039</td>
<td>STATUS_INVALID_STREAM_DESCRIPTION_VERSION</td>
<td>The tag count is not zero, but the tags are empty. Ensure that the tags are specified or the count is zero.</td>
<td></td>
</tr>
<tr>
<td>0x5200003a</td>
<td>STATUS_INVALID_STREAM_DESCRIPTION_VERSION</td>
<td>Specify the correct current version of the structure.</td>
<td></td>
</tr>
<tr>
<td>0x5200003b</td>
<td>STATUS_INVALID_TAG_NAME</td>
<td>Refer to the limits for the tag name that are specified in Producer SDK Limits (p. 44).</td>
<td></td>
</tr>
<tr>
<td>0x5200003c</td>
<td>STATUS_INVALID_TAG_VALUE</td>
<td>Refer to the limits for the tag value that are specified in Producer SDK Limits (p. 44).</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
<td>Recommended Action</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x5200003d</td>
<td>STATUS_TAG_STREAM_CALL_FAILED</td>
<td>The TagResource API failed.</td>
<td>The TagResource API call failed. Check for a valid network connection. See the logs for more information about the failure.</td>
</tr>
<tr>
<td>0x5200003e</td>
<td>STATUS_INVALID_CUSTOM_DATA</td>
<td>Invalid custom data has been specified in a call to the PIC APIs.</td>
<td>Invalid custom data has been specified in a call to the PIC APIs. This can occur only in the clients that directly use PIC.</td>
</tr>
<tr>
<td>0x5200003f</td>
<td>STATUS_INVALID_CREATE_STREAM_RESPONSE</td>
<td>Invalid CreateStreamResponse structure.</td>
<td>The structure or its member fields are invalid (that is, the ARN is null or larger than what's specified in Producer SDK Limits (p. 44)).</td>
</tr>
<tr>
<td>0x52000040</td>
<td>STATUS_CLIENT_AUTH_CALL_FAILED</td>
<td>Client auth failed.</td>
<td>The PIC failed to get proper auth information (that is, 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.</td>
</tr>
<tr>
<td>0x52000041</td>
<td>STATUS_GET_CLIENT_TOKEN_Call_FAILED</td>
<td>Getting the security token call failed.</td>
<td>This situation can occur for clients that use PIC directly. After a number of retries, the call fails with this error.</td>
</tr>
<tr>
<td>0x52000042</td>
<td>STATUS_CLIENT_PROVISION_FAILED</td>
<td>Provisioning error.</td>
<td>Provisioning is not implemented.</td>
</tr>
<tr>
<td>0x52000043</td>
<td>STATUS_CREATE_CLIENT_CALL_FAILED</td>
<td>Failed to create the producer client.</td>
<td>A generic error returned by the PIC after a number of retries when the client creation fails.</td>
</tr>
<tr>
<td>0x52000044</td>
<td>STATUS_CLIENT_READY_CALLBACK_FAILED</td>
<td>Failed to get the producer client to a READY state.</td>
<td>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.</td>
</tr>
<tr>
<td>0x52000045</td>
<td>STATUS_TAG_CLIENT_CALL_FAILED</td>
<td>The TagResource for the producer client failed.</td>
<td>The TagResource API call failed for the producer client. See the logs for more information about the root cause.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
<td>Recommended Action</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------</td>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x52000046</td>
<td>STATUS_INVALID_CREATE_DEVICE</td>
<td>Device/Producer creation failed.</td>
<td>The higher-level SDKs (for example, C++ or Java) don't implement the device/producer creation API yet. Clients that use PIC directly can indicate a failure using the result notification.</td>
</tr>
<tr>
<td></td>
<td>RESPONSE_FAILED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x52000047</td>
<td>STATUS_ACK_TIMESTAMP_NOT_IN_VIEW_WINDOW</td>
<td>The timestamp of the received ACK is not in the view.</td>
<td>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.</td>
</tr>
<tr>
<td>0x52000048</td>
<td>STATUS_INVALID_FRAGMENT_ACK_VERSION</td>
<td>InvalidFragmentAck structure version.</td>
<td>Specify the correct current version of the FragmentAck structure.</td>
</tr>
<tr>
<td>0x52000049</td>
<td>STATUS_INVALID_TOKEN_EXPIRATION</td>
<td>Invalid security token expiration.</td>
<td>The security token expiration should have an absolute time stamp in the future that is greater than the current time stamp, with a grace period. For the limits for the grace period, see the Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x5200004a</td>
<td>STATUS_END_OF_STREAM</td>
<td>End of stream (EOS) indicator.</td>
<td>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.</td>
</tr>
<tr>
<td>0x5200004b</td>
<td>STATUS_DUPLICATE_STREAM_NAME</td>
<td>Duplicate stream name.</td>
<td>Multiple streams can't have the same stream name. Choose a unique name for the stream.</td>
</tr>
<tr>
<td>0x5200004c</td>
<td>STATUS_INVALID_RETENTION_PERIOD</td>
<td>Invalid retention period.</td>
<td>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 (p. 44).</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
<td>Recommended Action</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x5200004d</td>
<td>STATUS_INVALID_ACK_KEY_NAME</td>
<td>Failed to parse the fragment ACK string. Invalid key start indicator.</td>
<td>The fragment ACK string might be damaged. It can self-correct and this error can</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FragmentAck.</td>
<td>be treated as a warning.</td>
</tr>
<tr>
<td>0x5200004e</td>
<td>STATUS_INVALID_ACK_DUP_KEY_NAME</td>
<td>Failed to parse the fragment ACK string. Multiple keys have the same name.</td>
<td>The fragment ACK string might be damaged. It can self-correct and this error can</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FragmentAck.</td>
<td>be treated as a warning.</td>
</tr>
<tr>
<td>0x5200004f</td>
<td>STATUS_INVALID_ACK_INVALID_VALUE_START</td>
<td>Failed to parse the fragment ACK string because of an invalid key value</td>
<td>The fragment ACK string might be damaged. It can self-correct and this error can</td>
</tr>
<tr>
<td></td>
<td></td>
<td>start indicator.</td>
<td>be treated as a warning.</td>
</tr>
<tr>
<td>0x52000050</td>
<td>STATUS_INVALID_ACK_INVALID_VALUE_END</td>
<td>Failed to parse the fragment ACK string because of an invalid key value</td>
<td>The fragment ACK string might be damaged. It can self-correct and this error can</td>
</tr>
<tr>
<td></td>
<td></td>
<td>end indicator.</td>
<td>be treated as a warning.</td>
</tr>
<tr>
<td>0x52000051</td>
<td>STATUS_INVALID_PARSED_ACK_TYPE</td>
<td>Failed to parse the fragment ACK string because an invalid ACK type was</td>
<td>The fragment ACK string might be damaged. It can self-correct and this error can</td>
</tr>
<tr>
<td></td>
<td></td>
<td>specified.</td>
<td>be treated as a warning.</td>
</tr>
<tr>
<td>0x52000052</td>
<td>STATUS_STREAM_HAS_BEEN_STOPPED</td>
<td>The stream has been stopped, but a frame is still being put into the stream.</td>
<td></td>
</tr>
<tr>
<td>0x52000053</td>
<td>STATUS_INVALID_STREAM_METRICS_VERSION</td>
<td>Specify the correct current version of the StreamMetrics structure.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>StreamMetrics structure version.</td>
<td></td>
</tr>
<tr>
<td>0x52000054</td>
<td>STATUS_INVALID_CLIENT_METRICS_VERSION</td>
<td>Specify the correct current version of the ClientMetrics structure.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ClientMetrics structure version.</td>
<td></td>
</tr>
<tr>
<td>0x52000055</td>
<td>STATUS_INVALID_CLIENT_INITIALIZATION</td>
<td>Failed to reach the READY state during the producer client initialization.</td>
<td>See the logs for more information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>failed to reach a READY state.</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
<td>Recommended Action</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x52000056</td>
<td>STATUS_STATE_MACHINE_STATE</td>
<td>Internal state machine error.</td>
<td>Not a publicly visible error.</td>
</tr>
<tr>
<td></td>
<td>NOT_FOUND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x52000057</td>
<td>STATUS_INVALID_FRAGMENT_ACK</td>
<td>Invalid ACK type is specified in the FragmentAck structure.</td>
<td>The FragmentAck structure should contain ACK types defined in the public header.</td>
</tr>
<tr>
<td>0x52000058</td>
<td>STATUS_INVALID_STREAM_STATE</td>
<td>Internal state machine transition error.</td>
<td>Not a publicly visible error.</td>
</tr>
<tr>
<td>0x52000059</td>
<td>STATUS_CLIENT_FREED_BEFORE_STREAM</td>
<td>The stream object was freed after the producer was freed.</td>
<td>There was an attempt to free a stream object after the producer object was freed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This can only occur in clients that directly use PIC.</td>
</tr>
<tr>
<td>0x5200005a</td>
<td>STATUS_ALLOCATION_SIZE_SMALLER_THAN_REQUESTED</td>
<td>Internal error indicating that the actual allocation size from the content store is smaller than the size of the packaged frame/fragment.</td>
<td></td>
</tr>
<tr>
<td>0x5200005b</td>
<td>STATUS_VIEW_ITEM_SIZE_GREATER_THAN_ALLOCATION</td>
<td>The stored size of the allocation in the content view is greater than the allocation size in the content store.</td>
<td></td>
</tr>
<tr>
<td>0x5200005c</td>
<td>STATUS_ACK_ERR_STREAM_READ</td>
<td>Stream read error ACK.</td>
<td>An error that the ACK returned from the backend indicating a stream read/parsing error. This generally occurs when the backend fails to retrieve the stream. Auto-restreaming can usually correct this error.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
<td>Recommended Action</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x5200005d</td>
<td>STATUS_ACK_ERR_FRAGMENT_SIZE_REACHED</td>
<td>The maximum fragment size was reached.</td>
<td>The max fragment size in bytes is defined in Producer SDK Limits (p. 44). 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 ensure 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.</td>
</tr>
<tr>
<td>0x5200005e</td>
<td>STATUS_ACK_ERR_FRAGMENT_DURATION_REACHED</td>
<td>The maximum fragment duration was reached.</td>
<td>The max fragment duration is defined in Producer SDK Limits (p. 44). 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 ensure that key frames are being produced properly at the regular intervals.</td>
</tr>
<tr>
<td>0x5200005f</td>
<td>STATUS_ACK_ERR_CONNECTION_DURATION_REACHED</td>
<td>The maximum connection duration was reached.</td>
<td>Kinesis Video Streams enforces the max connection duration as specified in the Producer SDK Limits (p. 44). The Producer SDK automatically rotates the stream/token before the maximum is reached, and so clients using the SDK should not receive this error.</td>
</tr>
<tr>
<td>0x52000060</td>
<td>STATUS_ACK_ERR_FRAGMENT_TIMECODE_NOT_MONOTONIC</td>
<td>Timecodes are not monotonically increasing.</td>
<td>The Producer SDK enforces time stamps, so clients using the SDK should not receive this error.</td>
</tr>
<tr>
<td>0x52000061</td>
<td>STATUS_ACK_ERR_MULTI_TRACK_MKV</td>
<td>Multiple tracks in the MKV.</td>
<td>The Producer SDK enforces single track streams, so clients using the SDK should not receive this error.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
<td>Recommended Action</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x52000062</td>
<td>STATUS_ACK_ERR_INVALID_MKV_DATA</td>
<td>Invalid MKV data.</td>
<td>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 or 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/resolution, increase the compression ratio, or (in the case of a “bursty” network) allow for larger content store and buffer duration to accommodate for the temporary pressures.</td>
</tr>
<tr>
<td>0x52000063</td>
<td>STATUS_ACK_ERR_INVALID_PRODUCER_TIMESTAMP</td>
<td>Invalid producer time stamp.</td>
<td>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. Ensure that the system clock is set properly. Clients using the PIC directly should ensure that their callback functions return the correct time stamp.</td>
</tr>
<tr>
<td>0x52000064</td>
<td>STATUS_ACK_ERR_STREAM_NOT_ACTIVE</td>
<td>Inactive stream.</td>
<td>A call to a backend API was made while the stream was not in an &quot;Active&quot; 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.</td>
</tr>
<tr>
<td>0x52000065</td>
<td>STATUS_ACK_ERR_KMS_KEY_ACCESS_DENIED</td>
<td>AWS KMS access denied error.</td>
<td>Returned when the account has no access to the specified key.</td>
</tr>
<tr>
<td>0x52000066</td>
<td>STATUS_ACK_ERR_KMS_KEY_DISABLED</td>
<td>AWS KMS key is disabled</td>
<td>The specified key has been disabled.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
<td>Recommended Action</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------</td>
<td>-------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x52000067</td>
<td>STATUS_ACK_ERR_KMS_KEY_ERROR</td>
<td>AWS KMS key validation error.</td>
<td>Generic validation error. For more information, see the AWS Key Management Service API Reference.</td>
</tr>
<tr>
<td>0x52000068</td>
<td>STATUS_ACK_ERR_KMS_KEY_UNAVAIL</td>
<td>AWS KMS key unavailable.</td>
<td>The key is unavailable. For more information, see the AWS Key Management Service API Reference.</td>
</tr>
<tr>
<td>0x52000069</td>
<td>STATUS_ACK_ERR_KMS_KEY_INVALID</td>
<td>invalid use of AWS KMS key.</td>
<td>The AWS KMS key is not configured to be used in this context. For more information, see the AWS Key Management Service API Reference.</td>
</tr>
<tr>
<td>0x5200006a</td>
<td>STATUS_ACK_ERR_KMS_KEY_INVALID</td>
<td>AWS KMS invalid state.</td>
<td>For more information, see the AWS Key Management Service API Reference.</td>
</tr>
<tr>
<td>0x5200006b</td>
<td>STATUS_ACK_ERR_KMS_KEY_NOT_FOUND</td>
<td>AWS KMS key not found.</td>
<td>The key was not found. For more information, see the AWS Key Management Service API Reference.</td>
</tr>
<tr>
<td>0x5200006c</td>
<td>STATUS_ACK_ERR_STREAM_DELETED</td>
<td>The stream has been or is being deleted.</td>
<td>The stream is being deleted by another application or through the AWS Management Console.</td>
</tr>
<tr>
<td>0x5200006d</td>
<td>STATUS_ACK_ERR_ACK_INTERNAL</td>
<td>Internal error.</td>
<td>Generic service internal error.</td>
</tr>
<tr>
<td>0x5200006e</td>
<td>STATUS_ACK_ERR_FRAGMENT_ARCHIVAL</td>
<td>Fragment archival error.</td>
<td>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.</td>
</tr>
<tr>
<td>0x5200006f</td>
<td>STATUS_ACK_ERR_UNKNOWN</td>
<td>Unknown error.</td>
<td>The service returned an unknown error.</td>
</tr>
<tr>
<td>0x52000070</td>
<td>STATUS_MISSING_ERR_ACK</td>
<td>Missing ACK information.</td>
<td>The ACK parser completed parsing, but the FragmentAck information is missing.</td>
</tr>
<tr>
<td>0x52000071</td>
<td>STATUS_INVALID_ACK_SEGMENT_LENGTH</td>
<td>Invalid ACK segment length.</td>
<td>An ACK segment string with an invalid length was specified to the ACK parser. For more information, see Producer SDK Limits (p. 44).</td>
</tr>
</tbody>
</table>
## Error and Status Codes Returned by the Duration Library

The following table contains error and status information that is returned by methods in the **Duration** library.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFFFFFFFFFFFFFFFF</td>
<td>INVALID_DURATION_VALUE</td>
</tr>
</tbody>
</table>

## Error and Status Codes Returned by the Common Library

The following table contains error and status information that is returned by methods in the **Common** library.

**Note**

These error and status information codes are common to many APIs.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000001</td>
<td>STATUS_NULL_ARG</td>
<td>NULL was passed for a mandatory argument.</td>
</tr>
<tr>
<td>0x00000002</td>
<td>STATUS_INVALID_ARG</td>
<td>An invalid value was specified for an argument.</td>
</tr>
<tr>
<td>0x00000003</td>
<td>STATUS_INVALID_ARG_LEN</td>
<td>An invalid argument length was specified.</td>
</tr>
<tr>
<td>0x00000004</td>
<td>STATUS_NOT_ENOUGH_MEMORY</td>
<td>Could not allocate enough memory.</td>
</tr>
<tr>
<td>0x00000005</td>
<td>STATUS_BUFFER_TOO_SMALL</td>
<td>The specified buffer size is too small.</td>
</tr>
<tr>
<td>0x00000006</td>
<td>STATUS_UNEXPECTED_EOF</td>
<td>An unexpected end of file was reached.</td>
</tr>
<tr>
<td>0x00000007</td>
<td>STATUS_FORMAT_ERROR</td>
<td>An invalid format was encountered.</td>
</tr>
<tr>
<td>0x00000008</td>
<td>STATUS_INVALID_HANDLE_ERROR</td>
<td>Invalid handle value.</td>
</tr>
<tr>
<td>0x00000009</td>
<td>STATUS_OPEN_FILE_FAILED</td>
<td>Failed to open a file.</td>
</tr>
<tr>
<td>0x0000000a</td>
<td>STATUS_READ_FILE_FAILED</td>
<td>Failed to read from a file.</td>
</tr>
<tr>
<td>0x0000000b</td>
<td>STATUS_WRITE_TO_FILE_FAILED</td>
<td>Failed to write to a file.</td>
</tr>
<tr>
<td>0x0000000c</td>
<td>STATUS_INTERNAL_ERROR</td>
<td>An internal error that normally doesn't occur and might indicate an SDK or service API bug.</td>
</tr>
<tr>
<td>0x0000000d</td>
<td>STATUS_INVALID_OPERATION</td>
<td>There was an invalid operation, or the operation is not permitted.</td>
</tr>
<tr>
<td>0x0000000e</td>
<td>STATUS_NOT_IMPLEMENTED</td>
<td>The feature is not implemented.</td>
</tr>
<tr>
<td>0x0000000f</td>
<td>STATUS_OPERATION_TIMED_OUT</td>
<td>The operation timed out.</td>
</tr>
</tbody>
</table>
## Error Code Reference

### Code | Message | Description
--- | --- | ---
0x00000010 | STATUS_NOT_FOUND | A required resource was not found.

### Error and Status Codes Returned by the Heap Library

The following table contains error and status information that is returned by methods in the Heap library.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01000001</td>
<td>STATUS_HEAP_FLAGS_ERROR</td>
<td>An invalid combination of flags was specified.</td>
</tr>
<tr>
<td>0x01000002</td>
<td>STATUS_HEAP_NOT_INITIALIZED</td>
<td>An operation was attempted before the heap was initialized.</td>
</tr>
<tr>
<td>0x01000003</td>
<td>STATUS_HEAP_CORRUPTED</td>
<td>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.</td>
</tr>
<tr>
<td>0x01000004</td>
<td>STATUS_HEAP_VRAM_LIB_MISSING</td>
<td>The VRAM (video RAM) user or kernel mode library cannot be loaded or is missing. Check if the underlying platform supports VRAM allocations.</td>
</tr>
<tr>
<td>0x01000005</td>
<td>STATUS_HEAP_VRAM_LIB_REOPEN</td>
<td>Failed to open the VRAM library.</td>
</tr>
<tr>
<td>0x01000006</td>
<td>STATUS_HEAP_VRAM_INIT_FUNC_SYMBOL</td>
<td>Failed to load the INIT function export.</td>
</tr>
<tr>
<td>0x01000007</td>
<td>STATUS_HEAP_VRAMALLOC_FUNC_SYMBOL</td>
<td>Failed to load the ALLOC function export.</td>
</tr>
<tr>
<td>0x01000008</td>
<td>STATUS_HEAP_VRAMFREE_FUNC_SYMBOL</td>
<td>Failed to load the FREE function export.</td>
</tr>
<tr>
<td>0x01000009</td>
<td>STATUS_HEAP_VRAM_LOCK_FUNC_SYMBOL</td>
<td>Failed to load the LOCK function export.</td>
</tr>
<tr>
<td>0x0100000a</td>
<td>STATUS HEAP_VRAM_UNLOCK_FUNC_SYMBOL</td>
<td>Failed to load the UNLOCK function export.</td>
</tr>
<tr>
<td>0x0100000b</td>
<td>STATUS_HEAP_VRAM_UNINIT_FUNC_SYMBOL</td>
<td>Failed to load the UNINIT function export.</td>
</tr>
<tr>
<td>0x0100000c</td>
<td>STATUS_HEAP_VRAM_GETMAX_FUNC_SYMBOL</td>
<td>Failed to load the GETMAX function export.</td>
</tr>
<tr>
<td>0x0100000d</td>
<td>STATUS_HEAP_DIRECT_MEM_INIT</td>
<td>Failed to initialize the main heap pool in the hybrid heap.</td>
</tr>
<tr>
<td>0x0100000e</td>
<td>STATUS HEAP_VRAM_INIT_FAILED</td>
<td>The VRAM dynamic initialization failed.</td>
</tr>
</tbody>
</table>
### Error and Status Codes Returned by the MKVGen Library

The following table contains error and status information that is returned by methods in the MKVGen library.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description / Recommended Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0100000f</td>
<td>STATUS_HEAP_LIBRARY_FREE_FAILED</td>
<td>Failed to de-allocate and free the VRAM library.</td>
</tr>
<tr>
<td>0x01000010</td>
<td>STATUS_HEAP_VRAM_ALLOC_FAILED</td>
<td>The VRAM allocation failed.</td>
</tr>
<tr>
<td>0x01000011</td>
<td>STATUS_HEAP_VRAM_FREE_FAILED</td>
<td>The VRAM free failed.</td>
</tr>
<tr>
<td>0x01000012</td>
<td>STATUS_HEAP_VRAM_MAP_FAILED</td>
<td>The VRAM map failed.</td>
</tr>
<tr>
<td>0x01000013</td>
<td>STATUS_HEAP_VRAM_UNMAP_FAILED</td>
<td>The VRAM unmap failed.</td>
</tr>
<tr>
<td>0x01000014</td>
<td>STATUS_HEAP_VRAM_UNINIT_FAILED</td>
<td>The VRAM deinitialization failed.</td>
</tr>
<tr>
<td>0x32000001</td>
<td>STATUS_MKV_INVALID_FRAME_DATA</td>
<td>Invalid members of the Frame data structure. Ensure that the duration, size, and frame data are valid and are within the limits specified in Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x32000002</td>
<td>STATUS_MKV_INVALID_FRAME_TIMESTAMP</td>
<td>Invalid frame time stamp. The calculated PTS (presentation time stamp) and DTS (decoding time stamp) are greater or equal to the time stamp 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” (p. 102)</td>
</tr>
<tr>
<td>0x32000003</td>
<td>STATUS_MKV_INVALID_CLUSTER_DURATION</td>
<td>An invalid fragment duration was specified. For more information, see Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x32000004</td>
<td>STATUS_MKV_INVALID_CONTENT</td>
<td>Invalid content type string length. For more information, see Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x32000005</td>
<td>STATUS_MKV_NUMBER_TOO_BIG</td>
<td>There was an attempt to encode a number that's too large to be represented in EBML (Extensible Binary Meta Language) format.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description / Recommended Action</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x32000006</td>
<td>STATUS_MKV_INVALID_CODEC_ID_LENGTH</td>
<td>Invalid codec ID string length. For more information, see Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x32000007</td>
<td>STATUS_MKV_INVALID_TRACK_NAME_LENGTH</td>
<td>Invalid track name string length. For more information, see Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x32000008</td>
<td>STATUS_MKV_INVALID_CODEC_PRIVATE_LENGTH</td>
<td>Invalid codec private data length. For more information, see Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x32000009</td>
<td>STATUS_MKV_CODEC_PRIVATE_NULL</td>
<td>The codec private data (CPD) is NULL, whereas the CPD size is greater than 0.</td>
</tr>
<tr>
<td>0x3200000a</td>
<td>STATUS_MKV_INVALID_TIMECODE_SCALE</td>
<td>Invalid timecode scale value. For more information, see Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x3200000b</td>
<td>STATUS_MKV_MAX_FRAME_TIMECODE</td>
<td>The frame timecode is greater than the maximum. For more information, see Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x3200000c</td>
<td>STATUS_MKV_LARGE_FRAME_TIMECODE</td>
<td>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 cannot 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.</td>
</tr>
<tr>
<td>0x3200000d</td>
<td>STATUS_MKV_INVALID_ANNEB_NALU_IN_FRAME_DATA</td>
<td>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 &quot;emulation prevention&quot; sequence to escape a sequence of three or more zeroes in the bytestream. For more information, see the MPEG specification.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description / Recommended Action</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x3200000e</td>
<td>STATUS_MKV_INVALID_AVCC_NALU_IN_FRAME_DATA</td>
<td>Invalid AVCC NALu packaging when the adapting AVCC flag is specified. Ensure that the bytestream is in a valid AVCC format. For more information, see the MPEG specification.</td>
</tr>
<tr>
<td>0x3200000f</td>
<td>STATUS_MKV_BOTH_ANNEXB_AND_AVCC_SPECIFIED</td>
<td>Both adapting AVCC and Annex-B NALs were specified. Specify either one, or specify none.</td>
</tr>
<tr>
<td>0x32000010</td>
<td>STATUS_MKV_INVALID_ANNEXB_NALU_IN_CPD</td>
<td>Invalid Annex-B format of CPD when the adapting Annex-B flag is specified. Ensure that the CPD is in valid Annex-B format. If it is not, then remove the CPD Annex-B adaptation flag.</td>
</tr>
<tr>
<td>0x32000011</td>
<td>STATUS_MKV_PTS_DTS_ARE_NOT_SAME</td>
<td>Kinesis Video Streams enforces the PTS (presentation time stamp) and DTS (decoding time stamp) to be the same for the fragment start frames. These are the key frames that start the fragment.</td>
</tr>
<tr>
<td>0x32000012</td>
<td>STATUS_MKV_INVALID_H264_H265_CPD</td>
<td>Failed to parse H264/H265 codec private data.</td>
</tr>
<tr>
<td>0x32000013</td>
<td>STATUS_MKV_INVALID_H264_H265_SPS_WIDTH</td>
<td>Failed to extract the width from the codec private data.</td>
</tr>
<tr>
<td>0x32000014</td>
<td>STATUS_MKV_INVALID_H264_H265_SPS_HEIGHT</td>
<td>Failed to extract the height from the codec private data.</td>
</tr>
<tr>
<td>0x32000015</td>
<td>STATUS_MKV_INVALID_H264_H265_SPS_NALU</td>
<td>Invalid H264/H265 SPS NALu.</td>
</tr>
<tr>
<td>0x32000016</td>
<td>STATUS_MKV_INVALID_BIH_CPD</td>
<td>Invalid bitmap info header format in the codec private data.</td>
</tr>
</tbody>
</table>

### Error and Status Codes Returned by the Trace Library

The following table contains error and status information that is returned by methods in the Trace library.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10100001</td>
<td>STATUS_MIN_PROFILER_BUFFER</td>
</tr>
</tbody>
</table>

### Error and Status Codes Returned by the Utils Library

The following table contains error and status information that is returned by methods in the Utils library.
### Error Code Reference

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x40000001</td>
<td>STATUS_INVALID_BASE64_ENCODE</td>
</tr>
<tr>
<td>0x40000002</td>
<td>STATUS_INVALID_BASE</td>
</tr>
<tr>
<td>0x40000003</td>
<td>STATUS_INVALID_DIGIT</td>
</tr>
<tr>
<td>0x40000004</td>
<td>STATUS_INT_OVERFLOW</td>
</tr>
<tr>
<td>0x40000005</td>
<td>STATUS_EMPTY_STRING</td>
</tr>
<tr>
<td>0x40000006</td>
<td>STATUS_DIRECTORY_OPEN_FAILED</td>
</tr>
<tr>
<td>0x40000007</td>
<td>STATUS_PATH_TOO_LONG</td>
</tr>
<tr>
<td>0x40000008</td>
<td>STATUS_UNKNOWN_DIR_ENTRY_TYPE</td>
</tr>
<tr>
<td>0x40000009</td>
<td>STATUS_REMOVE_DIRECTORY_FAILED</td>
</tr>
<tr>
<td>0x4000000a</td>
<td>STATUS_REMOVE_FILE_FAILED</td>
</tr>
<tr>
<td>0x4000000b</td>
<td>STATUS_REMOVE_LINK_FAILED</td>
</tr>
<tr>
<td>0x4000000c</td>
<td>STATUS_DIRECTORY_ACCESS_DENIED</td>
</tr>
<tr>
<td>0x4000000d</td>
<td>STATUS_DIRECTORY_MISSING_PATH</td>
</tr>
<tr>
<td>0x4000000e</td>
<td>STATUS_DIRECTORY_ENTRY_STAT_ERROR</td>
</tr>
</tbody>
</table>

### Error and Status Codes Returned by the View Library

The following error and status information that is returned by methods in the `View` library:

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x30000001</td>
<td>STATUS_MIN_CONTENT_VIEW_ITEM_COUNT</td>
<td>An invalid content view item count was specified. For more information, see Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x30000002</td>
<td>STATUS_INVALID_CONTENT_VIEW_DURATION</td>
<td>An invalid content view duration was specified. For more information, see Producer SDK Limits (p. 44).</td>
</tr>
<tr>
<td>0x30000003</td>
<td>STATUS_CONTENT_VIEW_NO_MORE_ITEMS</td>
<td>An attempt was made to get past the head position.</td>
</tr>
<tr>
<td>0x30000004</td>
<td>STATUS_CONTENT_VIEW_INVALID_INDEX</td>
<td>An invalid index is specified.</td>
</tr>
<tr>
<td>0x30000005</td>
<td>STATUS_CONTENT_VIEW_INVALID_TIME_STAMP</td>
<td>An invalid time stamp or a time stamp overlap. The frame decoding time stamp should be greater or equal to the previous frame time stamp, plus the previous frame duration: <code>DTS(n) &gt;= DTS(n-1) +...</code></td>
</tr>
</tbody>
</table>
NAL Adaptation Flags

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x30000006</td>
<td>STATUS_INVALID_CONTENT_VIEW_LENGTH</td>
<td>An invalid content view item data length was specified.</td>
</tr>
</tbody>
</table>

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:
### Producer Structures

This section includes information on structures used to provide data to the Producer SDK.

#### Topics
- DeviceInfo/ DefaultDeviceInfoProvider (p. 66)
- StorageInfo (p. 66)

#### 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 ensure the correct version of the structure is used with the current version of the codebase. The current version is specified with 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 the device can handle. This pre-allocates the storage for pointers to the stream objects initially, but the actual stream objects will be created later. The default is 16 streams, but this can be changed in the DefaultDeviceInfoProvider.cpp file.
- **storagelna**: An object describing the main storage configuration. See StorageInfo (p. 66).

#### StorageInfo

The StorageInfo object 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.

---

<table>
<thead>
<tr>
<th>Flag</th>
<th>Adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAL_ADAPTATION_FLAG_NONE</td>
<td>No adaptation</td>
</tr>
<tr>
<td>NAL_ADAPTATION_ANNEXB_NALS</td>
<td>Adapt Annex-B NALUs to AVCC NALUs</td>
</tr>
<tr>
<td>NAL_ADAPTATION_AVCC_NALS</td>
<td>Adapt AVCC NALUs to Annex-B NALUs</td>
</tr>
<tr>
<td>NAL_ADAPTATION_ANNEXB_CPD_NALS</td>
<td>Adapt Annex-B NALUs for the codec private data to AVCC format NALUs</td>
</tr>
<tr>
<td>NAL_ADAPTATION_ANNEXB_CPD_FRAME_NALS</td>
<td>Adapt Annex-B NALUs for the codec and frame private data to AVCC format NALUs</td>
</tr>
</tbody>
</table>

For more information about NALU types, see Section 1.3: Network Abstraction Layer Unit Types in RFC 3984.
As the memory is used, the virtual pages will be backed by the physical pages. This will result in low-memory pressure on the overall system when storage is underutilized.

The default storage size should be calculated based on the following formula:

\[
\text{Size} = \text{NumberOfStreams} \times \text{AverageFrameSize} \times \text{FramesPerSecond} \times \text{BufferDurationInSeconds} \times \text{DefragmentationFactor}
\]

In the preceding formula, the \text{DefragmentationFactor} should be set to 1.2 (20 percent).

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, and each stream has 3 minutes of buffer duration:

\[
\text{Size} = (512 \times 100 \times (3 \times 60) + 25 \times 10000 \times (3 \times 60)) \times 1.2 = (921600 + 4500000) \times 1.2 = 65059200 \approx 66\text{MB}.
\]

If the device has more available memory, it is prudent to add more memory to storage, to avoid severe fragmentation.

Care should be taken to ensure 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 during times when the bandwidth is low. If the producer is hitting memory pressure, it will emit storage overflow pressure callbacks (\text{StorageOverflowPressureFunc}), but when there is no memory available in the content store, it will drop the frame that's being pushed into KVS with an error (\text{STATUS_STORE_OUT_OF_MEMORY} = 0x5200002e, see Error and Status Codes Returned by the Client Library (p. 47) for more details.) This can also happen if the application ACKs are not available or the Persisted ACKs are delayed; in this case the buffers will fill to the “buffer duration” capacity before the older frames start dropping out.

**Member Fields**

- **version**: An integer value used to ensure the correct version of the structure is used with the current version of the codebase.
- **storageType**: A \text{DEVICE_STORAGE_TYPE} enumeration that specifies the underlying backing/implementation of the storage. Currently the only supported value is \text{DEVICE_STORAGE_TYPE_IN_MEM}. \text{DEVICE_STORAGE_TYPE_HYBRID_FILE} will be supported in a future implementation, indicating that storage falls back to the file-backed content store.
- **storageSize**: The storage size in bytes to pre-allocate. The minimum allocation is 10MB, and the maximum allocation is 10GB (This will change with the future implementation of the file backed content store).
- **spillRatio**: An integer value representing 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 stored. Not currently used.

**KVS Stream Structures**

This section includes information on structures used to provide data to an instance of a Kinesis video stream.

**StreamDefinition/ StreamInfo**

The \text{StreamDefinition} object in the C++ layer wraps the \text{StreamInfo} object in the Platform Independent Code (PIC), and provides some default values in the constructor.
## Member Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream_name</td>
<td>string</td>
<td>An optional stream name. See [Producer SDK Limits](p. 44) for more information about the length of the stream name. Each stream should have a unique name.</td>
<td>If no name is specified, a name will be generated randomly.</td>
</tr>
<tr>
<td>retention_period</td>
<td>duration&lt;uint64_t, ratio&lt;3600&gt;&gt;</td>
<td>The retention period for the stream in seconds. Specifying 0 indicates no retention.</td>
<td>3600 (One hour)</td>
</tr>
<tr>
<td>tags</td>
<td>const map&lt;string, string&gt;*</td>
<td>A map of key/value pairs containing user information. If the stream has already a set of tags, the new tags will be appended to the existing set of tags.</td>
<td>No tags</td>
</tr>
<tr>
<td>kms_key_id</td>
<td>string</td>
<td>The KMS key id to be used for encrypting the stream. For more information, see [Using Server-Side Encryption with Kinesis Video Streams](p. 10).</td>
<td>The default KMS key (aws/kinesis-video)</td>
</tr>
<tr>
<td>streaming_type</td>
<td>STREAMING_TYPE enumeration</td>
<td>The only supported value is STREAMING_TYPE_REALTIME.</td>
<td></td>
</tr>
<tr>
<td>content_type</td>
<td>string</td>
<td>The content format of the stream. The KVS console can play back content in the video/h264 format.</td>
<td>video/h264</td>
</tr>
<tr>
<td>max_latency</td>
<td>duration&lt;uint64_t, milli&gt;</td>
<td>The maximum latency in milliseconds for the stream. The stream latency pressure callback (if specified) will be called when the buffer duration exceeds this amount of time. Specifying 0 indicates that no stream latency pressure callback will be called.</td>
<td>milliseconds::zero()</td>
</tr>
<tr>
<td>Field</td>
<td>Data Type</td>
<td>Description</td>
<td>Default Value</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>fragment_duration</td>
<td>duration&lt;uint64_t&gt;</td>
<td>The desired fragment duration in seconds. This value is used in combination with the key_frame_fragmentation value. If this value is false, then the KVS service will generate fragments on a key frame after this duration elapses. For example, an AAC audio stream has each frame as a key-frame. Specifying key_frame_fragmentation = false will cause fragmentation to happen on a key-frame after this duration expires, resulting in 2 second fragments.</td>
<td>2</td>
</tr>
<tr>
<td>timecode_scale</td>
<td>duration&lt;uint64_t, milli&gt;</td>
<td>The MKV timecode scale in milliseconds, which will specify 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), and as such, care should be taken to ensure the frame timecode can be represented with the given timecode scale. The default timecode scale value of 1ms ensures that the largest frame that can be represented is 32767ms = 32 seconds, which is over the maximum fragment duration specified in Kinesis Video Streams Limits (p. 97), which is 10 seconds.</td>
<td>1</td>
</tr>
<tr>
<td>Field</td>
<td>Data Type</td>
<td>Description</td>
<td>Default Value</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>key_frame_fragmentation</td>
<td>bool</td>
<td>Whether to produce fragments on a key-frame. If true, the SDK will produce a start of the fragment every time there is a key-frame. If false, the KVS service will wait for at least fragment_duration and will produce a new fragment on the key-frame following it.</td>
<td>true</td>
</tr>
<tr>
<td>frame_timecodes</td>
<td>bool</td>
<td>Whether to use frame timecodes or generate timestamps using the current time callback. Many encoders do not produce timestamps with the frames, so specifying false for this parameter will ensure the frames are timestamped as they are put into the KVS SDK.</td>
<td>true</td>
</tr>
<tr>
<td>absolute_fragment_times</td>
<td>bool</td>
<td>KVS utilizes MKV as its underlying packaging mechanism. The MKV specification is strict about frame timecodes being relative to the beginning of the cluster (fragment), but 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 will use the optional stream start timestamp and adjust the cluster timestamps. The service always stores the fragments with their absolute timestamps.</td>
<td>true</td>
</tr>
<tr>
<td>Field</td>
<td>Data Type</td>
<td>Description</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>fragment_acks</td>
<td>bool</td>
<td>Whether to receive application level fragment ACKs (acknowledgements) or not.</td>
<td>true, meaning that the SDK will receive the ACKs and act accordingly.</td>
</tr>
<tr>
<td>restart_on_error</td>
<td>bool</td>
<td>Whether to restart on specific errors.</td>
<td>true, meaning that the SDK will attempt to restart the streaming if any errors happen.</td>
</tr>
<tr>
<td>recalculate_metrics</td>
<td>bool</td>
<td>Whether to re-calculate the metrics. Each call to retrieve the metrics can recalculate those to get the latest &quot;running&quot; value, which might create a small CPU impact. This setting might need to be set to false on extremely low-power/footprint devices to spare the CPU cycles. It's not advised to use false for this value otherwise.</td>
<td>true</td>
</tr>
<tr>
<td>Field</td>
<td>Data Type</td>
<td>Description</td>
<td>Default Value</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| nal_adaptation_flags | uint32_t   | Specifies the NAL unit adaptation flags. If the bitstream being H264 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/consumers (read encode/decoders) utilize the Annex-B format, as it has some advantages, such as error recovery. Higher-level systems use the AvCC format; this is the default for MPEG, HLS, DASH, etc. The console playback uses the browser's MSE (media source extensions) to decode and play back the stream which uses AvCC format. For H264 (and for M-JPEG and H265), the SDK provides adaptation capabilities. Many elementary streams are in the following format:  

\[
\text{Ab(Sps) Ab(Pps) Ab(I-frame) Ab(P/B-frame) Ab(P/B-frame) ... Ab(Sps) Ab(Pps) Ab(I-frame) Ab(P/B-frame) Ab(P/B-frame)}
\]

In the preceding example, Ab is the Annex-B start code (001 or 0001). The Codec Private Data (CPD) in the case of H.264 is in the SPS and PPS parameters, and can be adapted to the AvCC format. Unless the media pipeline gives the CPD separately, the default is to adapt Annex-B format to AvCC format for both the frame data and for the Codec Private Data. |
<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Application can extract the CPD from the frame by looking for the first Idr frame (which should contain the SPS/PPS), extract the two NALUs which will be Ab(Sps)Ab(Pps) and set it in the CPD in StreamDefinition. For more information about NAL adaptation flags, see NAL Adaptation Flags (p. 65).</td>
<td></td>
</tr>
<tr>
<td>frame_rate</td>
<td>uint32_t</td>
<td>The expected frame rate. This value is used to better calculate buffering needs.</td>
<td>25</td>
</tr>
<tr>
<td>avg_bandwidth_bps</td>
<td>int32_t</td>
<td>The expected average bandwidth for the stream. This value is used to better calculate buffering needs.</td>
<td>4 * 1024 * 1024</td>
</tr>
<tr>
<td>Field</td>
<td>Data Type</td>
<td>Description</td>
<td>Default Value</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>buffer_duration</td>
<td>duration&lt;uint64_t&gt;</td>
<td>The stream buffer duration in seconds. The SDK will keep the frames in the content store for up to the buffer_duration, after which the older frames will be dropped as the window moves forward. If the frame being dropped has not been sent to the backend, the dropped frame callback will be called. If the current buffer duration is greater than max_latency, then the stream latency pressure callback will be called. The buffer will be trimmed to the next fragment start when the Fragment Persisted ACK is received, indicating that the content has been durably persisted in the cloud, so storing the content on the local device is no longer needed.</td>
<td>120</td>
</tr>
<tr>
<td>Field</td>
<td>Data Type</td>
<td>Description</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>replay_duration</td>
<td>duration&lt;uint64_t&gt;</td>
<td>The duration to roll the current reader backward to replay during an error if restarting is enabled, in seconds. The rollback will stop at the buffer start (in case it has just started streaming, or the persisted ACK has come along). The rollback will attempt to land on a key frame that indicates a fragment start. If the error causing the restart is not indicative of a dead host (that is, the host is still alive and contains the frame data in its internal buffers) then the rollback will stop at the last Received ACK frame and roll forward to the next keyframe, as the entire fragment is already stored in the host memory.</td>
<td>40</td>
</tr>
<tr>
<td>connection_staleness</td>
<td>duration&lt;uint64_t&gt;</td>
<td>The time in seconds after which the stream staleness callback will be called, if the SDK does not receive the buffering ack. This is an indication that the frames are being sent from the device, but the backend is not acknowledging them, which is indicative of a severed connection at the intermediate hop or at the load balancer.</td>
<td>30</td>
</tr>
<tr>
<td>codec_id</td>
<td>string</td>
<td>The codec ID for the MKV track.</td>
<td>V_MPEG4/ISO/AVC</td>
</tr>
<tr>
<td>track_name</td>
<td>string</td>
<td>The MKV track name.</td>
<td>kinesis_video</td>
</tr>
</tbody>
</table>
### Stream Structures

<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>codecPrivateData</td>
<td>unsigned char*</td>
<td>The Codec Private Data (CPD) buffer. If the media pipeline has the information about the CPD before the stream starts, it can be set in StreamDefinition.codecPrivateData. The bits are copied, and the buffer can be re-used or freed after the call to create the stream. If, however, the data is not available at the stream creation time, it can be set in one of the overloads of the KinesisVideoStream.start(cpd) function.</td>
<td>null</td>
</tr>
<tr>
<td>codecPrivateDataSize</td>
<td>uint32_t</td>
<td>The Codec Private Data buffer size.</td>
<td>0</td>
</tr>
</tbody>
</table>

### ClientMetrics

The ClientMetrics object is filled by calling getKinesisVideoMetrics.

#### Member Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>UINT32</td>
<td>The version of the structure, defined in the CLIENT_METRICS_CURRENT_VERSION macro.</td>
</tr>
<tr>
<td>contentStoreSize</td>
<td>UINT64</td>
<td>The overall content store size in bytes. This is the value specified in DeviceInfo.StorageInfo.storageSize.</td>
</tr>
<tr>
<td>contentStoreAvailableSize</td>
<td>UINT64</td>
<td>Currently available storage size in bytes.</td>
</tr>
<tr>
<td>contentStoreAllocatedSize</td>
<td>UINT64</td>
<td>Currently allocated size. The allocated + available sizes should be slightly smaller than the overall storage size, due to the internal bookkeeping and the implementation of the content store.</td>
</tr>
<tr>
<td>totalContentViewsSize</td>
<td>UINT64</td>
<td>The size of the memory allocated for all content views.</td>
</tr>
</tbody>
</table>
Stream Structures

The **StreamMetrics** object is filled by calling `getKinesisVideoMetrics`.

**Member Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>version</strong></td>
<td>UINT32</td>
<td>The version of the structure, defined in the <code>STREAM_METRICS_CURRENT_VERSION</code> macro.</td>
</tr>
<tr>
<td><strong>currentViewDuration</strong></td>
<td>UINT64</td>
<td>The duration of the accumulated frames. In the fast networking case, this duration will be either 0 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 is specified. The duration is specified in 100ns units, which is the default time unit for the PIC layer.</td>
</tr>
<tr>
<td><strong>overallViewDuration</strong></td>
<td>UINT64</td>
<td>The overall view duration. If the stream is configured with no ACKs or Persistence, then this value will grow as the frames are put into the KVS stream, and will become 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, as the</td>
</tr>
<tr>
<td>Field</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ACK timestamp</td>
<td></td>
<td>ACK timestamp indicates the beginning of the entire fragment. The duration is specified in 100ns units, which is the default time unit for the PIC layer.</td>
</tr>
<tr>
<td>currentViewSize</td>
<td>UINT64</td>
<td>The size in bytes of the current buffer.</td>
</tr>
<tr>
<td>overallViewSize</td>
<td>UINT64</td>
<td>The overall view size in bytes.</td>
</tr>
<tr>
<td>currentFrameRate</td>
<td>UINT64</td>
<td>The observed frame rate for the current stream.</td>
</tr>
<tr>
<td>currentTransferRate</td>
<td>UINT64</td>
<td>The observed transfer rate in bytes per second for the current stream.</td>
</tr>
</tbody>
</table>
Kinesis Video Stream Parser Library

The Kinesis Video Stream Parser Library is an easy-to-use set of tools you can use in Java applications to consume the MKV data in a Kinesis video stream.

The library includes the following tools:

- **StreamingMkvReader (p. 80):** This class reads specified MKV elements from a video stream.
- **FragmentMetadataVisitor (p. 81):** This class retrieves metadata for fragments (media elements) and tracks (individual data streams containing media information, such as audio or subtitles).
- **OutputSegmentMerger (p. 82):** This class merges consecutive fragments or chunks in a video stream.
- **KinesisVideoExample (p. 83):** 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” (p. 80)
- the section called “Step 2: Write and Examine the Code” (p. 80)
- the section called “Step 3: Run and Verify the Code” (p. 85)

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, do the following:
  - Open [https://aws.amazon.com/](https://aws.amazon.com/), and then choose Create an AWS Account.
  - Follow the online instructions.
  - Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.
  - Note your AWS account ID because you need it for configuring programmatic access to Kinesis video streams.
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 (p. 79).

1. Create a directory and clone the library 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, 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” (p. 80)

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** (p. 80)
- **FragmentMetadataVisitor** (p. 81)
- **OutputSegmentMerger** (p. 82)
- **KinesisVideoExample** (p. 83)

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 StreamingMkvReader to retrieve MkvElement objects from an input stream called inputStream:

```java
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:

```java
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 (p. 80) 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. That is, 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:

```java
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`:

```java
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 FragmentMetaDataVisitor (p. 81) to 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 (p. 80) to parse the returned fragments on the stream, and uses a FragmentMetadataVisitor (p. 81) to log the fragments.

Delete and recreate the stream

The following code example (from the StreamOps.java file) deletes a given Kinesis video stream:

```java
//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:

```java
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:

```java
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:

```java
GetMediaResult result = videoMedia.getMedia(new
    GetMediaRequest().withStreamName(streamName).withStartSelector(startSelector));
```

Parse the GetMedia result

This section describes how to use StreamingMkvReader (p. 80), FragmentMetadataVisitor (p. 81) 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 (p. 80) and uses it to parse the result from the GetMedia operation:

```java
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 (p. 80) retrieves MKVElement objects from the payload of the GetMedia result. In the next section, the elements are passed to a FragmentMetadataVisitor (p. 81).

Retrieve Fragments with FragmentMetadataVisitor

The following code examples (from the KinesisVideoExample.java and StreamingMkvReader.java files) create a FragmentMetadataVisitor (p. 81). The MkvElement objects iterated by the StreamingMkvReader (p. 80) are then passed to the visitor using the accept method.

```java
from KinesisVideoExample.java:

FragmentMetadataVisitor fragmentMetadataVisitor = FragmentMetadataVisitor.create();
```

```java
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" (p. 82) that merges consecutive elements in the GetMedia result with the same track and EBML data.
- A CompositeMkvElementVisitor that composes the FragmentMetadataVisitor (p. 81), the section called "OutputSegmentMerger" (p. 82), and LogVisitor into a single element visitor

```java
//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 executes 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” (p. 85)

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 Kinesis Video Streams Developer Guide has following code examples:

- PutMedia API Example (p. 86)
- GStreamer Example (p. 89)
- Renderer Example (p. 90)
- KinesisVideoExample (p. 83)

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). So if you haven't already done so, you must 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

PutMedia API Example

This example demonstrates how to use the PutMedia API. It shows how to send data that is already in a container format (MKV). If your data needs to 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 (p. 22).

Note
- The PutMedia operation is available only in the C++ and Java SDKs, due to the full-duplex management of connections, data flow, and acknowledgements. It is not supported in other languages.

This example includes the following steps:

- Step 1: Download and Configure the Code (p. 86)
- Step 2: Write and Examine the Code (p. 87)
- Step 3: Run and Verify the Code (p. 89)

Step 1: Download and Configure the Code

In this section, you 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 (p. 23).

# git clone https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-java
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, 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, 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:
    2. In the JAR selection wizard, choose all .jars in the project's lib directory.

4. The PutMedia API example uses a profile named my-profile. To use another credentials profile, update the following line in PutMediaDemo.java:

```java
/* the AWS credentials profile configured by: aws configure --profile my-profile */
private static final String AWS_CREDENTIALS_PROFILE_NAME = "my-profile";
```

### Step 2: Write and Examine the Code

The PutMedia API example (PutMediaDemo) shows the following coding pattern:

#### Topics
- Create the PutMediaClient (p. 87)
- Stream Media and Pause the Thread (p. 88)

The code examples in this section are from the PutMediaDemo class.

### Create the PutMediaClient

Creating the PutMediaClient object takes 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 (p. 23) (my-stream). To use a different stream, change the following parameter:

```java
private static final String STREAM_NAME="my-stream";
```

**Note**
The PutMedia API example does not create a stream. You must create a stream either by using the test application for the Using the Java Producer Library (p. 23), by using the Kinesis Video Streams console, or by using the AWS CLI.

• The current time stamp.
• The time code type. The example uses RELATIVE, indicating that the time stamp 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:

```java
/* 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.

```java
/* start streaming video in a background thread */
client.putMediaInBackground();

/* wait for request/response to complete */
```
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. Choose PutMediaDemo.
3. Choose Run, Run 'PutMediaDemo'.

GStreamer Example

The C++ Producer Library (p. 31) contains a demo application for using GStreamer, an open source multimedia framework, with Amazon Kinesis Video Streams.

The following procedure demonstrates how to set up and use the GStreamer demo. The application sends the video data to Kinesis Video Streams on the following platforms:

- A built-in camera on a macOS computer
- A USB camera on a Linux or Raspberry Pi device

Note
The GStreamer example is not currently available for Windows systems.

Prerequisites

The GStreamer example typically requires the following components, which you can install using the following commands:

- Automake: brew install automake
- Autoconf: brew install autoconf automake
- CMake: brew install cmake
- GStreamer: brew install gstreamer gst-plugins-base gst-plugins-bad gst-plugins-good gst-plugins-ugly
- GNU Bison: brew install bison && brew link bison --force

For a more comprehensive list of requirements, see Prerequisites (p. 33) in the C++ Producer SDK documentation, or the README.md file in the SDK itself.

Running the GStreamer Example

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-cpp

2. Run the following script in the /kinesis-video-native-build directory to build the C++ Producer SDK:
This script installs the prerequisites for the SDK and builds the binaries, including the GStreamer example. For information about building the example on Ubuntu, see the "Troubleshooting" section of the README.md file in the SDK.

3. After the C++ Procedure SDK is installed and configured, run the application using the following command (also in the /kinesis-video-native-build directory):

```bash
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_DEFAULT_REGION=<AWS region>
./kinesis_video_gstreamer_sample_app stream_name
```

Supply the following information for the application parameters:

- **AWS_ACCESS_KEY_ID**: The access key ID for your account. See Prerequisites (p. 86).
- **AWS_SECRET_ACCESS_KEY**: The secret access key for your account. See Prerequisites (p. 86).
- **AWS_DEFAULT_REGION**: Your AWS Region, for example, us-west-2.
- **stream_name**: The name of the Kinesis video stream to send camera data to. The stream will be created if it doesn't exist.

4. The demo application starts. In a few seconds, you will see video data from your camera in the Kinesis Video Streams console for your stream.

   **Note**
   If the application fails to acquire your camera, you might see a Failed to negotiate pipeline error. For troubleshooting information, see the README.md file in the SDK.

## Renderer Example

The Stream Parser Library (p. 79) 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 GStreamer Example (p. 89) 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.

   ```bash
   $ 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.

   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 (p. 91)
- Parsing MKV Fragments into Frames (p. 91)
- Decoding and Displaying the Frame (p. 92)

### 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`:

```java
PutMediaWorker putMediaWorker = PutMediaWorker.create(getRegion(),
    getCredentialsProvider(),
    getStreamName(),
    inputStream,
    streamOps.amazonKinesisVideo);
executorService.submit(putMediaWorker);
```

For information about the `PutMediaWorker` class, see **Call PutMedia** (p. 83) in the Stream Parser Library (p. 79) 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 (p. 83) in the Stream Parser Library (p. 79) 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:

```java
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 Kinesis Video Streams

Monitoring is an important part of maintaining the reliability, availability, and performance of Kinesis Video Streams and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. Before you start monitoring Kinesis Video Streams, however, you should 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 have defined your monitoring goals and have created your monitoring plan, the next step is to establish a baseline for normal Kinesis Video Streams performance in your environment. You should measure Kinesis Video Streams performance at various times and under different load conditions. As you monitor Kinesis Video Streams, you should store a history of monitoring data that you've collected. You can compare current Kinesis Video Streams performance to this historical data to help you to identify normal performance patterns and performance anomalies, and devise methods to address issues that may arise.

Topics
• Monitoring Kinesis Video Streams Metrics with CloudWatch (p. 93)

Monitoring Kinesis Video Streams Metrics with CloudWatch

You can monitor a Kinesis video stream using Amazon CloudWatch, which collects and processes raw data from 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.

To access the CloudWatch dashboard for a Kinesis video stream, choose View stream metrics in CloudWatch in the Stream info section of the console page for the stream.

For a list of available metrics that Kinesis Video Streams supports, see Kinesis Video Streams Metrics and Dimensions.

CloudWatch Metrics Guidance

CloudWatch metrics can be useful for finding answers to the following questions:

Topics
• Is data reaching the Kinesis Video Streams service? (p. 94)
• Why is data not being successfully ingested by the Kinesis Video Streams service? (p. 94)
• Why can't the data be read from the Kinesis Video Streams service at the same rate as it's being sent from the producer? (p. 94)
• Why is there no video in the console, or why is the video being played with a delay? (p. 95)
• What is the delay in reading real-time data, and why is the client lagging behind the head of the stream? (p. 95)
• Is the client reading data out of the Kinesis video stream, and at what rate? (p. 95)
• Why can't the client read data out of the Kinesis video stream? (p. 96)

Is data reaching the Kinesis Video Streams service?

Relevant metrics:
• PutMedia.IncomingBytes
• PutMedia.IncomingFragments
• PutMedia.IncomingFrames

Action items:
• If there is 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 Kinesis Video Streams service?

Relevant metrics:
• PutMedia.Requests
• PutMedia.ConnectionErrors
• PutMedia.Success
• PutMedia.ErrorAckCount

Action items:
• If there is an increase in PutMedia.ConnectionErrors, look at the HTTP response/error codes received by the producer client to see what errors are occurring while establishing the connection.
• If there is 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 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 is 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 is 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 is a drop in PutMedia.Success, check the ack error codes. For more information, see AckErrorCode.Values.
• If there is an increase in PutMedia.FragmentPersistLatency or ListFragments.Latency, you are 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 is 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 is 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 is a drop in GetMedia.Success, then 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 is an increase in GetMedia.MillisBehindNow, look at your bandwidth limits to see if you are 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 what rate 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 is an increase in GetMedia.ConnectionErrors, look at the HTTP response/error codes returned by the GetMedia request. For more information, see AckErrorCode.Values.
• If you are trying to read the latest/live data, check PutMedia.IncomingBytes to see if there is data coming into the stream for the service to send to the consumers.
• If there is 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 is an issue with your service.
Kinesis Video Streams Limits

Kinesis Video Streams has the following limits:

The limits below are either soft [s], which can be upgraded by submitting a support ticket, or hard [h], which cannot be increased.

Control Plane API limits

The following section describes limits for Control Plane APIs.

When an account-level Request limit is reached, a ClientLimitExceededException is thrown.

When an account-level Streams limit is reached, or a stream-level limit is reached, a StreamLimitExceededException is thrown.

Control Plane API limits

<table>
<thead>
<tr>
<th>API</th>
<th>Account Limit: Request</th>
<th>Account Limit: Streams</th>
<th>Stream-level limit</th>
<th>Relevant Exceptions and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateStream</td>
<td>50 TPS [s]</td>
<td>100 streams per account [s]</td>
<td>5 TPS [h]</td>
<td>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.</td>
</tr>
<tr>
<td>DescribeStream</td>
<td>300 TPS [h]</td>
<td>N/A</td>
<td>5 TPS [h]</td>
<td></td>
</tr>
<tr>
<td>UpdateStream</td>
<td>50 TPS [h]</td>
<td>N/A</td>
<td>5 TPS [h]</td>
<td></td>
</tr>
<tr>
<td>ListStreams</td>
<td>300 TPS [h]</td>
<td>N/A</td>
<td>5 TPS [h]</td>
<td></td>
</tr>
<tr>
<td>DeleteStream</td>
<td>50 TPS [h]</td>
<td>N/A</td>
<td>5 TPS [h]</td>
<td></td>
</tr>
<tr>
<td>GetDataEndpoint</td>
<td>300 TPS [h]</td>
<td>N/A</td>
<td>5 TPS [h]</td>
<td>When combined with account limit, this implies a maximum of 60 streams can be Put to and Read from (with 4 consumers).</td>
</tr>
</tbody>
</table>

Media and Archived Media API limits

The following section describes limits for Media and Archived Media APIs.

When a stream-level limit is exceeded, a StreamLimitExceededException 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 minumum 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 limits

<table>
<thead>
<tr>
<th>API</th>
<th>Stream-level limit</th>
<th>Connection-level limit</th>
<th>Bandwidth limit</th>
<th>Fragment-level limit</th>
<th>Relevant Exceptions and Notes</th>
</tr>
</thead>
</table>
| PutMedia      | 5 TPS [h]          | 1 [s]                  | 12.5 MB/second, or 100 Mbps [s] | • Minimum fragment duration: 1 second [h]  
• Maximum fragment duration: 10 seconds [h]  
• Maximum fragment size: 50 MB [h] | A typical PutMedia request will contain data for several seconds, resulting in a lower TPS per stream. In the case of multiple concurrent connections that exceed limits, the last connection is accepted. |
| GetMedia      | 5 TPS [h]          | 3 [s]                  | 25 MB/s or 200 Mbps [s] | N/A                   | Only three clients can concurrently receive content from the media stream at any moment of time. Further client connections are rejected. A unique consuming client shouldn’t need more than 2 or 3 TPS, since once the connection is established, we anticipate that the application will read continuously. If a typical fragment is approximately 5 MB, this limit will mean ~75 MB/sec per Kinesis video stream. Such a stream would have an outgoing bit rate of 2x the streams' maximum incoming bit rate. |
| ListFragments | 5 TPS [h]          | 5 [s]                  | N/A                   | N/A                   | Five fragment-based consuming applications can concurrently list fragments based on processing requirements. |
| GetMediaForFragmentList | 5 TPS [h] | 5 [s] | 25 MB/s or 200 MbpsA [s] | Maximum number of fragments: 1000 [h] | Five fragment-based consuming applications can concurrently get media. Further connections are rejected. |
Troubleshooting Kinesis Video Streams

Use the following information to troubleshoot common issues encountered with Amazon Kinesis Video Streams.

Topics
- Troubleshooting General Issues (p. 99)
- Troubleshooting API Issues (p. 99)
- Troubleshooting Java Issues (p. 100)
- Troubleshooting Producer Library Issues (p. 101)
- Troubleshooting Stream Parser Library Issues (p. 105)

Troubleshooting General Issues

This section describes general issues that you might encounter when working with Kinesis Video Streams.

Issues
- Latency too high (p. 99)

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, and thus reduce the amount of time for each fragment, reduce the following value in kinesis_video_gstreamer_sample_app.cpp:

```c
G_OBJECT_SET(G_OBJECT(data.encoder), "bframes", 0, "key-int-max", 45, "bitrate", 512, NULL);
```

Troubleshooting API Issues

This section describes API issues that you might encounter when working with Kinesis Video Streams.

Issues
- Error: "Unable to determine service/operation name to be authorized" (p. 100)
- Error: "Failed to put a frame in the stream" (p. 100)
- Error: "Service closed connection before final AckEvent was received" (p. 100)
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 are 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. Ensure that the unread() methods are correctly implemented.

Troubleshooting Java Issues

This section describes how to troubleshoot common Java issues encountered when working with Kinesis Video Streams.

Issues

• Enabling Java logs (p. 101)
Enabling Java logs

To troubleshoot issues with Java samples and libraries, it is 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:

   ```xml
   <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:

   ```properties
   # 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.

Troubleshooting Producer Library Issues

This section describes issues that you might encounter when using the Producer Libraries (p. 22).

Issues

- Cannot compile the Producer SDK (p. 102)
- Video stream does not appear in the console (p. 102)
- Error: "Security token included in the request is invalid" when streaming data using the GStreamer demo application (p. 102)
- Error: "Failed to submit frame to Kinesis Video client" (p. 102)
- GStreamer application stops with "streaming stopped, reason not-negotiated" message on OS X (p. 103)
- Error: "Failed to allocate heap" when creating Kinesis Video Client in GStreamer demo on Raspberry Pi (p. 103)
- Error: "Illegal Instruction" when running GStreamer demo on Raspberry Pi (p. 103)
- Camera fails to load on Raspberry Pi (p. 103)
- Camera can't be found on macOS High Sierra (p. 104)
- jni.h file not found when compiling on macOS High Sierra (p. 104)
- Curl errors when running the GStreamer demo app (p. 104)
- Time stamp/range assertion at run time on Raspberry Pi (p. 104)
- Assertion on gst_value_set_fraction_range_full on Raspberry Pi (p. 104)
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 (p. 65) 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) which in the case of 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/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 time stamps 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 bit rate, 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:

```plaintext
```

A possible workaround for this is to remove the framerate parameters from the `gst_caps_new_simple` call in `kinesis_video_gstreamer_sample_app.cpp`:

```c
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,
    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`:

```c
device_info.storageInfo.storageSize = 512 * 1024 * 1024;
```

Error: "Illegal Instruction" when running GStreamer demo on Raspberry Pi

If you encounter the following error when executing the GStreamer demo, ensure that you have compiled the application for the correct version of your device. (For example, ensure that you are not compiling for Raspberry Pi 3 when you are running on Raspberry Pi 2.)

```plaintext
INFO - Initializing curl.
Illegal instruction
```

Camera fails to load on Raspberry Pi

To check whether the camera is loaded, run the following:

```bash
# ls /dev/video*
```

If nothing is found, run the following:

```bash
# 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 app

To resolve curl errors when you run the GStreamer demo application, copy this certificate file to /etc/ssl/cert.pem.

Time stamp/range assertion at run time on Raspberry Pi

If a time stamp range assertion occurs at run time, 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.

Troubleshooting Stream Parser Library Issues

This section describes issues that you might encounter when using the Stream Parser Library (p. 79).

Issues
- Cannot access a single frame from the stream (p. 105)
- Fragment decoding error (p. 105)

Cannot access a single frame from the stream

To access a single frame from a streaming source in your consumer application, ensure that your stream contains the correct codec private data. For information about the format of the data in a stream, see Data Model (p. 13).

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:

<table>
<thead>
<tr>
<th>Fragment Decoding Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>There was an error decoding the video data. Verify that the stream contains valid H.264 content</td>
</tr>
</tbody>
</table>

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 time stamps are accurate and in the correct order, and no duplicate time stamps are being created.
- Your application is encoding the frame data using the H.264 format.
# 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:** May 7, 2018

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Producer SDK structures reference documentation</td>
<td>Reference documentation for the structures used by the Kinesis Video Streams Producer Libraries (p. 22). For more information, see Producer SDK Structures (p. 66) and KVS Stream Structures (p. 67).</td>
<td>May 7, 2018</td>
</tr>
<tr>
<td>Renderer example documentation</td>
<td>Documentation for the Renderer example application, which shows how to decode and display frames from a Kinesis video stream. For more information, see Renderer Example (p. 90).</td>
<td>Mar 15, 2018</td>
</tr>
<tr>
<td>Producer SDK Limits reference documentation</td>
<td>Information about limits for operations in the C++ Producer Library (p. 31). For more information, see Producer SDK Limits (p. 44).</td>
<td>Mar 13, 2018</td>
</tr>
<tr>
<td>C++ Producer SDK for Raspberry Pi</td>
<td>Procedure for setting up and running the C++ Producer Library (p. 31) on a Raspberry Pi device. For more information, see Using the C++ Producer SDK on Raspberry Pi (p. 39).</td>
<td>Mar 13, 2018</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Information about monitoring Kinesis Video Streams metrics and API calls using Amazon CloudWatch and AWS CloudTrail. For more information, see Monitoring Kinesis Video Streams (p. 93).</td>
<td>February 5, 2018</td>
</tr>
<tr>
<td>Network Abstraction Layer (NAL) adaptation flag reference</td>
<td>Information about setting NAL adaptation flags when consuming streaming video.</td>
<td>January 15, 2018</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Android support for streaming video</td>
<td>Kinesis Video Streams now supports streaming video from Android devices. For more information, see Android Producer Library (p. 27).</td>
<td>January 12, 2018</td>
</tr>
<tr>
<td>Kinesis Video example documentation</td>
<td>Documentation for the Kinesis Video example application, which shows how to use the Kinesis Video Stream Parser Library (p. 79) in an application. For more information, see KinesisVideoExample (p. 83).</td>
<td>January 9, 2018</td>
</tr>
<tr>
<td>GStreamer example documentation</td>
<td>Documentation for the GStreamer example application that is included in the C++ Producer SDK. For more information, see GStreamer Example (p. 89).</td>
<td>January 5, 2018</td>
</tr>
<tr>
<td>Kinesis Video Streams documentation released</td>
<td>This is the initial release of the Amazon Kinesis Video Streams Developer Guide.</td>
<td>November 29, 2017</td>
</tr>
</tbody>
</table>
API Reference

This section contains the API Reference documentation.

Actions

The following actions are supported by Amazon Kinesis Video Streams:

- CreateStream (p. 109)
- DeleteStream (p. 113)
- DescribeStream (p. 115)
- GetDataEndpoint (p. 118)
- ListStreams (p. 121)
- ListTagsForStream (p. 124)
- TagStream (p. 127)
- UntagStream (p. 130)
- UpdateDataRetention (p. 132)
- UpdateStream (p. 135)

The following actions are supported by Amazon Kinesis Video Streams Media:

- GetMedia (p. 138)
- PutMedia (p. 142)

The following actions are supported by Amazon Kinesis Video Streams Archived Media:

- GetMediaForFragmentList (p. 148)
- ListFragments (p. 151)

Amazon Kinesis Video Streams

The following actions are supported by Amazon Kinesis Video Streams:

- CreateStream (p. 109)
- DeleteStream (p. 113)
- DescribeStream (p. 115)
- GetDataEndpoint (p. 118)
- ListStreams (p. 121)
- ListTagsForStream (p. 124)
- TagStream (p. 127)
- UntagStream (p. 130)
- UpdateDataRetention (p. 132)
- UpdateStream (p. 135)
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"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

DataRetentionInHours (p. 109)

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 (p. 109)

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
Pattern: [a-zA-Z0-9_.-]+
Required: No

**KmsKeyId (p. 109)**

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.

Required: No

**MediaType (p. 109)**

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.

To play video on the console, the media must be H.264 encoded, and you need to specify this video type in this parameter as video/h264.

This parameter is optional; the default value is null (or empty in JSON).

Type: String


Pattern: [\w\-\.\+]+/[\w\-\.\+]+

Required: No

**StreamName (p. 109)**

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

**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 (p. 110)

The Amazon Resource Name (ARN) of the stream.

Type: String


Pattern: \barn:aws:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+\b

Errors

For information about the errors that are common to all actions, see Common Errors (p. 162).

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

The stream is currently not available for this operation.

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
• AWS SDK for Java
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V2
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 (p. 113)

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 (p. 113)

The Amazon Resource Name (ARN) of the stream that you want to delete.
Type: String
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 (p. 162).

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
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2
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 (p. 115)

The Amazon Resource Name (ARN) of the stream.

Type: String


Pattern: `arn:aws:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`

Required: No

StreamName (p. 115)

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 (p. 115)

An object that describes the stream.

Type: StreamInfo (p. 155) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 162).

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
- AWS SDK for Java
- AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V2
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 (p. 118)**
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

Required: Yes

**StreamARN (p. 118)**
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


Pattern: `arn:aws:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9._-]+/[0-9]+`

Required: No

**StreamName (p. 118)**
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 (p. 119)

- 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 (p. 162).

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
• AWS SDK for Java
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V2
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 (p. 121)

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 (p. 121)

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.

Required: No

StreamNameCondition (p. 121)

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 (p. 157) 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 (p. 121)

   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.

StreamInfoList (p. 121)

   An array of StreamInfo objects.

   Type: Array of StreamInfo (p. 155) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 162).

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
• AWS SDK for Java
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V2
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 (p. 124)

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.

Required: No

StreamARN (p. 124)

The Amazon Resource Name (ARN) of the stream that you want to list tags for.

Type: String


Pattern: `arn:aws:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9-_.-]+/[0-9]+`

Required: No

StreamName (p. 124)

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 (p. 125)**

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.

**Tags (p. 125)**

A map of tag keys and values associated with the specified stream.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 162).

**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**
- **AWS SDK for Java**
- **AWS SDK for JavaScript**
- **AWS SDK for PHP V3**
- **AWS SDK for Python**
- **AWS SDK for Ruby V2**
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 User Guide.

You must provide either the StreamName or the StreamARN.

This operation requires permission for the KinesisVideo:TagStream action.

Kinesis video streams support up to 50 tags.

Request Syntax

```plaintext
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 (p. 127)

The Amazon Resource Name (ARN) of the resource that you want to add the tag or tags to.

Type: String


Pattern: arn:aws:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9-_.]+/[0-9]+

Required: No

StreamName (p. 127)

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 (p. 127)

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

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

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 (p. 162).

**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. Kinesis video streams 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
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2
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 (p. 130)

The Amazon Resource Name (ARN) of the stream that you want to remove tags from.

- Type: String
- Pattern: arn:aws:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-zA-Z0-9\-_\.]+/[a-z0-9\-_]+:[0-9]+
- Required: No

StreamName (p. 130)

The name of the stream that you want to remove tags from.

- Type: String
- Pattern: [a-zA-Z0-9\-_]+:
- Required: No

TagKeyList (p. 130)

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.
- 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 (p. 162).

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
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2
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**.

**Note**
The retention period that you specify replaces the current value.

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**

```json
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 (p. 132)**
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 (p. 132)**
The retention period, in hours. The value you specify replaces the current value.

Type: Integer
Valid Range: Minimum value of 1.
Required: Yes

Operation (p. 132)
Indicates whether you want to increase or decrease the retention period.
Type: String
Valid Values: INCREASE_DATA_RETENTION | DECREASE_DATA_RETENTION
Required: Yes

StreamARN (p. 132)
The Amazon Resource Name (ARN) of the stream whose retention period you want to change.
Type: String
Pattern: arn:aws:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9-_.-]+/[0-9]+
Required: No

StreamName (p. 132)
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 (p. 162).

ClientLimitExceeded Exception
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

The stream is currently not available for this operation.

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
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2
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 (p. 135)

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 (p. 135)

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


Pattern: [a-zA-Z0-9_\-.]+

Required: No
**MediaType (p. 135)**

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
Pattern: `[w\-\./\+]+/[w\-\./\+]+`
Required: No

**StreamARN (p. 135)**

The ARN of the stream whose metadata you want to update.

Type: String
Pattern: `arn:aws:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9_.-]+/[0-9]+`
Required: No

**StreamName (p. 135)**

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 (p. 162).

**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**

The stream is currently not available for this operation.

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
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

**Amazon Kinesis Video Streams Media**

The following actions are supported by Amazon Kinesis Video Streams Media:

- GetMedia (p. 138)
- PutMedia (p. 142)
GetMedia
Service: Amazon Kinesis Video Streams Media

Use this API to retrieve media content from a Kinesis video stream. In the request, you identify 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 to which you can then send the GetMedia requests.

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 (p. 142). 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.

Request Syntax

```json
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 (p. 138)

Identifies the starting chunk to get from the specified stream.

Type: StartSelector (p. 158) object

Required: Yes

StreamARN (p. 138)

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
Pattern: arn:aws:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9-_\.]+/[0-9]+
Required: No

StreamName (p. 138)
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: Content-Type

Response Elements
If the action is successful, the service sends back an HTTP 200 response.
The response returns the following HTTP headers.

ContentType (p. 139)
The content type of the requested media.
Pattern: ^[a-zA-Z0-9-_\.\-]+$

The response returns the following as the HTTP body.

Payload (p. 139)
The payload Kinesis Video Streams returns is a sequence of chunks from the specified stream. For information about the chunks, see PutMedia (p. 142). 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 time stamp of the fragment.
- AWS_KINESISVIDEO_PRODUCER_TIMESTAMP - Producer time stamp 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 (p. 162).

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

Status Code: 400, Caller used wrong endpoint to write data to a stream. On receiving such an exception, the user must call GetDataEndpoint with AccessMode set to "READ" and use the endpoint Kinesis Video returns in the next GetMedia call.

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

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
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2
**PutMedia**

Service: Amazon Kinesis Video Streams Media

Use this API to send media data to a Kinesis video stream.

**Note**

Before using this API, you must call the *GetDataEndpoint* API to get an endpoint. You then specify the endpoint in your *PutMedia* request.

In the request, you use the HTTP headers to provide parameter information, for example, stream name, time stamp, and whether the time stamp 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.
- 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. Note that 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 10 seconds and that contain more than 50 megabytes of data are not allowed.
- 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 - Time stamp when Kinesis Video Streams started receiving the fragment.
  - producer_timestamp - Time stamp, 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

\[ \text{producer_timestamp} = \text{producerStartTimestamp} + \text{fragment timecode} \]

If fragmentTimeCodeType is absolute, then

\[ \text{producer_timestamp} = \text{fragment timecode (converted to milliseconds)} \]

- 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.

**Request Syntax**

```http
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 requires the following URI parameters.

**FragmentTimecodeType (p. 143)**

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

**ProducerStartTimestamp (p. 143)**

You pass this value as the x-amzn-producer-start-timestamp HTTP header.
This is the producer time stamp at which the producer started recording the media (not the time stamp of the specific fragments in the request).

**StreamARN (p. 143)**

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`.


Pattern: `arn:aws:kinesisvideo:[a-z0-9-]+:[0-9]+:[a-z]+/[a-zA-Z0-9-_.-]+/[0-9]+$`

**StreamName (p. 143)**

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 (p. 143)**

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 (p. 144)**

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 codes:
  • 4000 - Error reading the data stream.
  • 4001 - Fragment size is greater than maximum limit, 50 MB, allowed.
  • 4002 - Fragment duration is greater than maximum limit, 10 seconds, allowed.
  • 4003 - Connection duration is greater than maximum allowed threshold.
  • 4004 - Fragment timecode is less than the timecode previous time code (within a **PutMedia** call, you cannot send fragments out of order).
  • 4005 - More than one track is found in MKV.
  • 4006 - Failed to parse the input stream as valid MKV format.
  • 4007 - Invalid producer timestamp.
  • 4008 - Stream no longer exists (deleted).
  • 4500 - Access to the stream's specified KMS key is denied.
  • 4501 - The stream's specified KMS key is disabled.
  • 4502 - The stream's specified KMS key failed validation.
  • 4503 - The stream's specified KMS key is unavailable.
  • 4504 - Invalid usage of the stream's specified KMS key.
  • 4505 - The stream's specified KMS key is in an invalid state.
  • 4506 - The stream's specified KMS key is not found.
  • 5000 - Internal service error
  • 5001 - 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 (p. 162).

**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**

Status Code: 400, Caller used wrong endpoint to write data to a stream. On receiving such an exception, the user must call GetDataEndpoint with AccessMode set to "READ" and use the endpoint Kinesis Video returns in the next GetMedia call.

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**

Status Code: 404, The stream with the given name does not exist.

HTTP Status Code: 404

Example

**Acknowledgement Format**

The format of the acknowledgement is as follows:

```json
{
    "Acknowledgement" : {
        "EventType": enum,
        "FragmentTimecode": Long,
        "FragmentNumber": Long,
        "ErrorId" : String
    }
}
```

---

146
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
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

Amazon Kinesis Video Streams Archived Media

The following actions are supported by Amazon Kinesis Video Streams Archived Media:

- GetMediaForFragmentList (p. 148)
- ListFragments (p. 151)
GetMediaForFragmentList
Service: Amazon Kinesis Video Streams Archived Media

Gets media for a list of fragments (specified by fragment number) from the archived data in a Kinesis video stream.

The following limits apply when using the GetMediaForFragmentList API:

- A client can call GetMediaForFragmentList 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 GetMediaForFragmentList session.

Request Syntax

```
POST /getMediaForFragmentList HTTP/1.1
Content-type: application/json

{
    "Fragments": [ "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 (p. 148)**

A list of the numbers of fragments for which to retrieve media. You retrieve these values with ListFragments (p. 151).

- Type: Array of strings
- Pattern: ^[0-9]+$
- Required: Yes

**StreamName (p. 148)**

The name of the stream from which to retrieve fragment media.

- Type: String
- Pattern: [a-zA-Z0-9_.-]+$
- Required: Yes

Response Syntax

```
HTTP/1.1 200
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The response returns the following HTTP headers.

**ContentType (p. 148)**

The content type of the requested media.


Pattern: `^[a-zA-Z0-9_\-\./]+$`

The response returns the following as the HTTP body.

**Payload (p. 148)**

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_KINESISVIDEOSERVER_SIDE_TIMESTAMP - Server-side time stamp of the fragment.
- AWS_KINESISVIDEO_PRODUCER_SIDE_TIMESTAMP - Producer-side time stamp 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 exception
- 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 (p. 162)](https://docs.aws.amazon.com/kinesis-video-streams/latest/devguide/AmazonKinesisVideoStreamsDeveloperGuide.html).

**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**

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

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
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2
ListFragments
Service: Amazon Kinesis Video Streams Archived Media

Returns a list of Fragment (p. 160) objects from the specified stream and start location within the archived data.

Request Syntax

POST /listFragments HTTP/1.1
Content-type: application/json

{
   "FragmentSelector": {
      "FragmentSelectorType": "string",
      "TimestampRange": {
         "EndTimestamp": number,
         "StartTimestamp": number
      }
   },
   "MaxResults": number,
   "NextToken": "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 (p. 151)

Describes the time stamp range and time stamp origin for the range of fragments to return.

Type: FragmentSelector (p. 161) object

Required: No

MaxResults (p. 151)

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 (p. 152) is provided in the output that you can use to resume pagination.

Type: Long

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken (p. 151)

A token to specify where to start paginating. This is the ListFragments:NextToken (p. 152) from a previously truncated response.

Type: String

Length Constraints: Minimum length of 1.

Required: No
StreamName (p. 151)

The name of the stream from which to retrieve a fragment list.

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

{
   "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 (p. 152)

A list of fragment numbers that correspond to the time stamp range provided.

Type: Array of Fragment (p. 160) objects

NextToken (p. 152)

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.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 162).

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

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

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
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

Data Types

The following data types are supported by Amazon Kinesis Video Streams:

- StreamInfo (p. 155)
- StreamNameCondition (p. 157)

The following data types are supported by Amazon Kinesis Video Streams Media:

- StartSelector (p. 158)

The following data types are supported by Amazon Kinesis Video Streams Archived Media:

- Fragment (p. 160)
- FragmentSelector (p. 161)
- TimestampRange (p. 162)

Amazon Kinesis Video Streams

The following data types are supported by Amazon Kinesis Video Streams:
- StreamInfo (p. 155)
- StreamNameCondition (p. 157)
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
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.
Required: No

MediaType
The MediaType of the stream.
Type: String
Pattern: [\\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


Pattern: `arn:aws: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-Z0-9-9]+`

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 Go
- AWS SDK for Java
- AWS SDK for Ruby V2
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 Go
• AWS SDK for Java
• AWS SDK for Ruby V2

Amazon Kinesis Video Streams Media

The following data types are supported by Amazon Kinesis Video Streams Media:

• StartSelector (p. 158)
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 time stamp (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

AfterFragmentNumber

Specifies the fragment number from where you want the GetMedia API to start returning the fragments.

Type: String
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
Pattern: ^[a-zA-Z0-9._-]+$
Required: No

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 containing the specific fragment. You must also specify the StartFragmentNumber.
- PRODUCER_TIMESTAMP or SERVER_TIMESTAMP - Start with the chunk containing a fragment with the specified producer or server time stamp. You specify the time stamp 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

**StartTimestamp**

A time stamp 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 time stamp.

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 Go
- AWS SDK for Java
- AWS SDK for Ruby V2

**Amazon Kinesis Video Streams Archived Media**

The following data types are supported by Amazon Kinesis Video Streams Archived Media:

- Fragment (p. 160)
- FragmentSelector (p. 161)
- TimestampRange (p. 162)
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 index value of the fragment.

Type: String
Length Constraints: Minimum length of 1.
Required: No

FragmentSizeInBytes

The total fragment size, including information about the fragment and contained media data.

Type: Long
Required: No

ProducerTimestamp

The time stamp from the producer corresponding to the fragment.

Type: Timestamp
Required: No

ServerTimestamp

The time stamp from the AWS server corresponding to the fragment.

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 Go
- AWS SDK for Java
- AWS SDK for Ruby V2
FragmentSelector
Service: Amazon Kinesis Video Streams Archived Media

Describes the time stamp range and time stamp origin of a range of fragments.

Contents

FragmentSelectorType

The origin of the time stamps to use (Server or Producer).

Type: String

Valid Values: PRODUCER_TIMESTAMP | SERVER_TIMESTAMP

Required: Yes

TimestampRange

The range of time stamps to return.

Type: TimestampRange (p. 162) 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 Go
- AWS SDK for Java
- AWS SDK for Ruby V2
**TimestampRange**
Service: Amazon Kinesis Video Streams Archived Media

The range of time stamps for which to return fragments.

**Contents**

**EndTimestamp**
- The ending time stamp in the range of time stamps for which to return fragments.
- Type: Timestamp
- Required: Yes

**StartTimestamp**
- The starting time stamp in the range of time stamps 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 Go
- AWS SDK for Java
- AWS SDK for Ruby V2

**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

**InvalidParameterCombination**

Parameters that must not be used together were used together.

HTTP Status Code: 400

**InvalidParameterValue**

An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

**InvalidQueryParameter**

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

**MalformedQueryString**

The query string contains a syntax error.

HTTP Status Code: 404

**MissingAction**

The request is missing an action or a required parameter.

HTTP Status Code: 400

**MissingAuthenticationToken**

The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

**MissingParameter**

A required parameter for the specified action is not supplied.

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 Signature Version 4 Signing Process in the Amazon Web Services General Reference.

**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 Task 2: Create a String to Sign for Signature Version 4 in the Amazon Web Services General Reference.

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 Handling Dates in Signature Version 4 in the Amazon Web Services General Reference.

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 Security Token Service, go to AWS Services That Work with IAM in the IAM User Guide.

Condition: If you’re using temporary security credentials from the AWS Security Token Service, 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 Task 1: Create a Canonical Request For Signature Version 4 in the Amazon Web Services General Reference.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional