# Table of Contents

What is Amazon IVS? .......................................................... 1  
What is Ultra-Low-Latency Live Video? .......................................................... 1  
Global Solution, Regional Control .......................................................... 1  
  Streaming and Viewing are Global .................................................. 1  
  Control is Regional ................................................................. 2  
  Your Channel’s Region .............................................................. 2  
Getting Started with Amazon IVS .................................................... 3  
  Step 1: Create an AWS Account .................................................. 3  
  Step 2: Set Up IAM Permissions .................................................. 3  
    Create a New User and Add Permissions .................................... 4  
    Add Permissions to an Existing User ......................................... 5  
  Step 3: Create a Channel with Optional Recording ....................... 6  
    Auto-Record to Amazon S3 ...................................................... 6  
    Console Instructions ............................................................... 7  
    CLI Instructions ...................................................................... 14  
  Step 4: Set Up Streaming Software ............................................. 16  
    Streaming with the Amazon IVS Broadcast SDK ..................... 17  
    Streaming with OBS Studio ..................................................... 17  
    Streaming a Recorded Video with FFmpeg ............................. 18  
  Step 5: View Your Live Stream .................................................... 18  
  Step 6: Check Your Service-Quota Limits (Optional) .................... 19  
    How to Disable Recording ....................................................... 19  
Broadcasting to Amazon IVS ....................................................... 20  
  Platform Requirements ................................................................ 20  
    Native Platforms .................................................................... 20  
  Required Device Access ............................................................. 20  
  Support ..................................................................................... 21  
    Versioning .............................................................................. 21  
Broadcasting to Amazon IVS: SDK for Android Guide ..................... 22  
  Getting Started ......................................................................... 22  
  Advanced Use Cases ................................................................. 24  
  Known Issues and Workarounds .................................................. 28  
Broadcasting to Amazon IVS: SDK for iOS Guide ............................ 28  
  Getting Started ......................................................................... 29  
  Advanced Use Cases ................................................................. 33  
  Known Issues and Workarounds .................................................. 36  
Amazon IVS Player ............................................................................ 38  
  Browser & Platform Requirements ............................................. 39  
    Desktop Browsers ................................................................... 39  
    Mobile Browsers ..................................................................... 39  
    Native Platforms ..................................................................... 40  
  Support ..................................................................................... 40  
    Versioning .............................................................................. 40  
Amazon IVS Player: SDK for Web Guide ............................................. 41  
  Getting Started ......................................................................... 41  
  Framework Integrations ............................................................. 42  
  Working With Content Security Policy ....................................... 43  
  Known Issues and Workarounds .................................................. 44  
Amazon IVS Player: SDK for Android Guide ..................................... 44  
  Getting Started ......................................................................... 45  
  SDK Size .................................................................................. 47  
  Known Issues and Workarounds .................................................. 47  
Amazon IVS Player: SDK for iOS Guide ............................................. 48  
  Getting Started ......................................................................... 48
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDK Size</td>
<td>52</td>
</tr>
<tr>
<td>Known Issues and Workarounds</td>
<td>52</td>
</tr>
<tr>
<td>Amazon IVS Player: Video.js Integration</td>
<td>53</td>
</tr>
<tr>
<td>Getting Started</td>
<td>53</td>
</tr>
<tr>
<td>Events</td>
<td>55</td>
</tr>
<tr>
<td>Errors</td>
<td>55</td>
</tr>
<tr>
<td>Plugins</td>
<td>56</td>
</tr>
<tr>
<td>Content Security Policy</td>
<td>56</td>
</tr>
<tr>
<td>Functions</td>
<td>56</td>
</tr>
<tr>
<td>currentTime</td>
<td>58</td>
</tr>
<tr>
<td>dispose</td>
<td>58</td>
</tr>
<tr>
<td>duration</td>
<td>59</td>
</tr>
<tr>
<td>getIVSEvents</td>
<td>59</td>
</tr>
<tr>
<td>getIVSPlayer</td>
<td>59</td>
</tr>
<tr>
<td>load</td>
<td>60</td>
</tr>
<tr>
<td>play</td>
<td>60</td>
</tr>
<tr>
<td>playbackRate</td>
<td>61</td>
</tr>
<tr>
<td>seekable</td>
<td>61</td>
</tr>
<tr>
<td>Amazon IVS Player: JW Player Integration</td>
<td>62</td>
</tr>
<tr>
<td>Getting Started</td>
<td>62</td>
</tr>
<tr>
<td>Events</td>
<td>63</td>
</tr>
<tr>
<td>Errors</td>
<td>64</td>
</tr>
<tr>
<td>Content Security Policy</td>
<td>64</td>
</tr>
<tr>
<td>Limitations</td>
<td>64</td>
</tr>
<tr>
<td>Embedding Metadata within a Video Stream</td>
<td>65</td>
</tr>
<tr>
<td>What is Timed Metadata?</td>
<td>65</td>
</tr>
<tr>
<td>Setting Up IAM Permissions</td>
<td>65</td>
</tr>
<tr>
<td>Inserting Timed Metadata</td>
<td>66</td>
</tr>
<tr>
<td>Using the AWS CLI</td>
<td>66</td>
</tr>
<tr>
<td>Using the Amazon IVS API</td>
<td>66</td>
</tr>
<tr>
<td>Consuming Timed Metadata</td>
<td>67</td>
</tr>
<tr>
<td>Sample Demo: Quiz App</td>
<td>67</td>
</tr>
<tr>
<td>Viewing Timed Metadata</td>
<td>68</td>
</tr>
<tr>
<td>For More Information</td>
<td>68</td>
</tr>
<tr>
<td>Setting Up Private Channels</td>
<td>69</td>
</tr>
<tr>
<td>Workflow for Private Channels</td>
<td>69</td>
</tr>
<tr>
<td>Create or Import a Playback Key</td>
<td>70</td>
</tr>
<tr>
<td>Console Instructions</td>
<td>70</td>
</tr>
<tr>
<td>CLI Instructions</td>
<td>71</td>
</tr>
<tr>
<td>API Request</td>
<td>71</td>
</tr>
<tr>
<td>Enable Playback Authorization on Channels</td>
<td>71</td>
</tr>
<tr>
<td>Console Instructions</td>
<td>71</td>
</tr>
<tr>
<td>CLI Instructions</td>
<td>72</td>
</tr>
<tr>
<td>API Request (Create and Update)</td>
<td>73</td>
</tr>
<tr>
<td>Generate and Sign Playback Tokens</td>
<td>73</td>
</tr>
<tr>
<td>Token Schema</td>
<td>73</td>
</tr>
<tr>
<td>Instructions</td>
<td>74</td>
</tr>
<tr>
<td>List Playback Keys</td>
<td>74</td>
</tr>
<tr>
<td>Console Instructions</td>
<td>74</td>
</tr>
<tr>
<td>CLI Instructions</td>
<td>74</td>
</tr>
<tr>
<td>API Request</td>
<td>75</td>
</tr>
<tr>
<td>Delete Playback Keys</td>
<td>75</td>
</tr>
<tr>
<td>Console Instructions</td>
<td>75</td>
</tr>
<tr>
<td>CLI Instructions</td>
<td>75</td>
</tr>
<tr>
<td>API Request</td>
<td>76</td>
</tr>
<tr>
<td>Get Information about Playback Keys</td>
<td>76</td>
</tr>
<tr>
<td>Console Instructions</td>
<td>76</td>
</tr>
</tbody>
</table>
Service Quotas Integration with CloudWatch Usage Metrics ........................................... 113
Creating a CloudWatch Alarm for Usage Metrics ....................................................... 114
Streaming Configuration ......................................................................................... 115
Prerequisites .......................................................................................................... 115
Reducing Latency ................................................................................................... 115
Avoid Third-Party Streaming/Forwarding Services ................................................. 116
Encoder Settings ................................................................................................... 116
  Stream Ingest: Codecs, RTMPs, and Port 443 ......................................................... 116
  Resolution/Bitrate/FPS ......................................................................................... 116
  Channel Types ..................................................................................................... 117
  Video Settings ...................................................................................................... 117
  Audio Settings ....................................................................................................... 117
  Use CBR, Not VBR ............................................................................................... 117
  Use Progressive Signals ....................................................................................... 117
Network Requirements ............................................................................................ 118
Closed Captioning .................................................................................................. 118
Stream from Windows Desktop with FFmpeg .......................................................... 119
Stream from Android and iOS with Larix Broadcaster ............................................. 119
Testing the Stream ................................................................................................. 120
Costs ....................................................................................................................... 121
Live Video .............................................................................................................. 121
Auto-Record to Amazon S3 .................................................................................. 121
  Storing Recorded Video ....................................................................................... 121
  Serving Recorded Video ....................................................................................... 123
Glossary ................................................................................................................... 124
Document History .................................................................................................... 126
  User Guide Changes ............................................................................................ 126
  API Reference Changes ...................................................................................... 139
Release Notes ......................................................................................................... 142
  September 8, 2021 ............................................................................................. 142
  Amazon IVS Player SDK 1.4.1 ............................................................................ 142
  August 13, 2021 ................................................................................................. 143
  ListTagsForResource API Endpoint ..................................................................... 143
  August 10, 2021 ................................................................................................. 143
  Amazon IVS Player SDK 1.4.0 ............................................................................ 143
  July 27, 2021 ..................................................................................................... 146
  Amazon IVS Broadcast SDK: Android 1.0.0 and iOS 1.0.0 ................................. 146
  June 1, 2021 ...................................................................................................... 146
  Amazon IVS Player SDK: Android 1.3.3 and iOS 1.3.3 .................................. 146
  May 19, 2021 ..................................................................................................... 147
  Amazon IVS Player SDK: Android 1.3.2 .......................................................... 147
  May 5, 2021 ........................................................................................................ 147
  Amazon IVS Player SDK 1.3 .............................................................................. 147
  April 26, 2021 .................................................................................................... 150
  Service Quotas Integration with CloudWatch Usage Metrics ............................. 150
  April 13, 2021 .................................................................................................... 151
  New CloudWatch Metrics .................................................................................. 151
  April 7, 2021 ...................................................................................................... 151
  Auto-Record to S3 (R2S3) ................................................................................ 151
  January 28, 2021 ............................................................................................... 151
  Amazon IVS Player SDK: JW Player Integration 1.2.0 .................................... 151
  December 16, 2020 ........................................................................................ 151
  Amazon IVS Player: SDK for Android 1.2.1 ................................................... 151
  November 23, 2020 ........................................................................................ 152
  Amazon IVS Player SDK 1.2.0 ......................................................................... 152
  November 12, 2020 ........................................................................................ 153
  New Event Field, stream_id ............................................................................. 153
Amazon Interactive Video Service User Guide
What is Ultra-Low-Latency Live Video?

What is Amazon Interactive Video Service?

Amazon Interactive Video Service (IVS) is a managed, live-video streaming service that allows you to:

- Create channels and start streaming in minutes.
- Build engaging, interactive experiences alongside ultra-low-latency live video.
- Distribute video at scale to a range of devices and platforms.
- Easily integrate into websites and apps.

Amazon IVS lets you focus on building your own interactive application and audience experience. With Amazon IVS, you don’t need to manage infrastructure or develop and configure components of your video workflows, to be secure, reliable, and cost effective.

Amazon IVS supports RTMPS streaming. RTMPS is the secure version of RTMP (Real-Time Messaging Protocol), running over TLS. RTMP is an industry standard for transmitting video over a network.

In addition to the product documentation here, see https://ivs.rocks/, a dedicated site to browse published content (demos, code samples, blog posts), estimate cost, and experience Amazon IVS through live demos.

What is Ultra-Low-Latency Live Video?

Low latency reduces the delay from when a camera captures a live stream to when the stream appears on a viewer’s screen. Amazon IVS can deliver video with latency under five seconds. For a traditional Over-The-Top (OTT) stream, latency may be as high as 30 seconds.

Low latency is a critical component in building good interactive user experiences that enrich the audience experience. It allows the streamer, the brand, and the community to connect with live audiences in a direct and personal way.

Observed latency can vary between users due to:

- The geographic locations of the streamer and viewers.
- Network type and speed.
- Individual components in the streaming chain.
- Streaming protocols and output formats.

For more information, see Reducing Latency in Amazon IVS Streaming Configuration.

Global Solution, Regional Control

Streaming and Viewing are Global

You can use Amazon IVS to stream to viewers worldwide:
When you stream, Amazon IVS automatically ingests video at a location near you. Viewers can watch your live streams globally via the Amazon IVS content-delivery network.

Another way of saying this is that the "data plane" is global. The data plane refers to streaming/ingesting and viewing.

**Control is Regional**

While the Amazon IVS data plane is global, the "control plane" is regional. The control plane refers to the Amazon IVS console, API, and resources (channels, stream keys, playback key pairs, and recording configurations).

Another way of saying this is that Amazon IVS is a "regional AWS service." That is, Amazon IVS resources in each region are independent of similar resources in other regions. For example, a channel that you create in one region is independent of channels you create in other regions.

When you use resources (e.g., create a channel), you must specify the region in which it will be created. Subsequently, when you manage resources, you must do so from the same region where they were created.

<table>
<thead>
<tr>
<th>If you use the ...</th>
<th>You specify the region by ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon IVS console</td>
<td>Using the <strong>Select a Region</strong> drop-down in the top right of the navigation bar.</td>
</tr>
<tr>
<td>Amazon IVS API</td>
<td>Using the appropriate service endpoint. See the Amazon IVS API Reference. (If you access the API through an SDK, set up the SDK's <strong>region</strong> parameter. See Tools to Build on AWS.)</td>
</tr>
</tbody>
</table>
| AWS CLI            | Either:  
  • Appending **--region <aws-region>** to your CLI command.  
  • Putting the region in your local AWS configuration file. |

*Remember, regardless of the region in which a channel was created, you can stream to Amazon IVS from anywhere, and viewers can watch from anywhere.*

**Your Channel’s Region**

Your channel's region is part of the ARN (Amazon Resource Name) that is assigned when you create the channel. When you create a channel:

- The Amazon IVS console shows the ARN in the **General configuration** area of the page. Subsequently, the console always shows your region (location) on the top right.
- The Amazon IVS API returns the ARN in the channel object's **channelArn** field.
Getting Started with Amazon Interactive Video Service

This document takes you through the steps to set up your first Amazon Interactive Video Service (IVS) live stream.

Topics
• Step 1: Create an AWS Account (p. 3)
• Step 2: Set Up IAM Permissions (p. 3)
• Step 3: Create a Channel with Optional Recording (p. 6)
• Step 4: Set Up Streaming Software (p. 16)
• Step 5: View Your Live Stream (p. 18)
• Step 6: Check Your Service-Quota Limits (Optional) (p. 19)
• How to Disable Recording (p. 19)

Step 1: Create an AWS Account

To use Amazon IVS, you need an AWS account. If you don't already have one, you are prompted to create it when you sign up. To create an AWS account:

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code. Also, you will have to provide billing information, although the basic tier of service is free. You are not charged for any AWS services that you sign up for unless you use them.

3. After creating the account, you will get one email with your Sign-in URL and User Name and another email (from your AWS account administrator) with your password. You must change the password during your first sign-in.

Step 2: Set Up IAM Permissions

Next, you must create an AWS Identity and Access Management (IAM) user and add a policy that gives the user access to create an Amazon IVS channel. If you want to auto-record to Amazon S3, you also must add appropriate permissions for that.

You can either add permissions in conjunction with creating a new user (p. 4) or add permissions to an existing user (p. 5). Both procedures are given below.

For more information (for example, to learn about IAM users and policies, how to attach a policy to a user, and how to constrain what users can do with Amazon IVS), see:

• Creating an IAM User in the IAM User Guide
• The IAM information in Amazon IVS Security (p. 100)
For record-to-S3 functionality: Using Service-Linked Roles (p. 107) and Auto-Record to Amazon S3 (p. 77) in the Amazon IVS User Guide

Create a New User and Add Permissions

Follow these steps:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Users, then choose Add user.
3. In the Add user window:
   a. Type the new user name to be created.
   b. Check Programmatic access and AWS Management Console access.
   c. Choose Next: Permissions.
4. Under Set Permissions, turn on Attach existing policies directly, then choose Create Policy. A Create Policy window opens.
5. In the Create Policy window, choose the JSON tab, and copy and paste the following IVS policy to the JSON tab. This policy covers both standard video and auto-record-to-S3 functionality.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
        ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": ["iam:CreateServiceLinkedRole", "iam:AttachRolePolicy", "iam:PutRolePolicy"
```
Add Permissions to an Existing User

Follow these steps:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Users, then choose an existing user name to be updated.
3. On the Summary page, on the Permissions tab, choose Add Permissions.
4. Under Add Permissions, turn on Attach existing policies directly, then choose Create Policy. A Create Policy window opens.
5. In the Create Policy window, choose the JSON tab, and copy and paste the following IVS policy to the JSON tab. This policy covers both standard video and auto-record-to-S3 functionality.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ivs:CreateChannel",
        "ivs:ListChannels",
        "ivs:GetChannel",
        "ivs:ListStreamKeys",
        "ivs:GetStreamKey",
        "ivs:ListStreams",
        "ivs:CreateRecordingConfiguration",
        "ivs:GetRecordingConfiguration",
        "ivs:ListRecordingConfigurations"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "servicequotas:ListServiceQuotas",
        "servicequotas:ListServices",
        "servicequotas:ListAWSDefaultServiceQuotas",
        "servicequotas:ListRequestedServiceQuotaChangeHistoryByQuota",
        "servicequotas:ListTagsForResource",
        "cloudwatch:GetMetricData",
        "cloudwatch:DescribeAlarms"
      ],
      "Resource": "*"
    }
  ]
}
```

7. On the Tags page, choose Review Policy. Give the policy a Name, then choose Create Policy. Then close this window and go back to the Add User window.
8. Back in the Add user window, attach the new policy. First refresh the policy table by pressing the reload button above the table on the right. Filter the policy by typing the name. When your policy appears in the list, select it and choose Next: Tags. On the Tags page, choose Next: Review.
10. The final Success screen contains your Access key ID, Secret access key, and Password. Store all of these for future reference. When you are done, choose Close.
Step 3: Create a Channel with Optional Recording

An Amazon IVS channel stores configuration information related to your live stream. You first create a channel and then contribute video to it using the channel's stream key to start your live stream.

As part of channel creation, the following items are assigned:

- An **ingest server** identifies a specific Amazon IVS component that receives the stream, along with an ingestion protocol (RTMPs).
- Amazon IVS assigns a **stream key** when you create a channel and uses it to authorize streaming. *Treat the stream key like a secret, since it allows anyone to stream to the channel.*
- A **playback URL** identifies the endpoint to start playback for a specific channel. This endpoint can be used globally. It automatically selects the best location from the Amazon IVS global content delivery network for a viewer to stream the video. (Note that Amazon IVS does not support custom domains for playback. *Do not proxy the playback URL with your own domain; that does not work and will cause issues.*

You can create a channel — with or without recording — through the Amazon IVS console or the AWS CLI. Channel creation and recording are discussed below.

Auto-Record to Amazon S3

You have the option of enabling recording for a channel. If the auto-record to S3 feature is enabled, all streams on the channel are recorded and saved to an Amazon S3 storage bucket that you own. Subsequently, the recording is available for on-demand playback.
Setting this up is an advanced option. By default, recording is disabled when a channel is created.

Before you can set up a channel for recording, you must create a recording-configuration. This is a resource which specifies an Amazon S3 location where the recorded streams for the channel are stored. You can create and manage recording configurations using the console or CLI; both procedures are given below. After you create the recording configuration, you associate it with a channel either when you create the channel (as described below) or later, by updating an existing channel. (In the API, see CreateChannel and UpdateChannel.) You can associate multiple channels with the same recording configuration. You can delete a recording configuration that is no longer associated with any channels.

Keep in mind the following constraints:

- You must own the S3 bucket. That is, the account that sets up a channel to be recorded must own the S3 bucket where recordings will be stored.
- The channel, recording configuration, and S3 location must be in the same AWS region. If you create channels in other regions and want to record them, you must also set up recording configurations and S3 buckets in those regions.

Recording to your S3 bucket requires authorization with your AWS credentials. To give IVS the required access, an AWS IAM Service-Linked Role (SLR) is created automatically when the recording configuration is created: the SLR is limited to give IVS write permission only on the specific bucket.

To disable recording after you have enabled it, see the section called “How to Disable Recording” (p. 19).

Note that network issues between the streaming location and AWS or within AWS could result in some data loss while recording your stream. In these cases, Amazon IVS prioritizes the live stream over the recording. For redundancy, record locally via your streaming tool.

For more information (including how to set up post-processing or VOD playback on your recorded files), see Auto-Record to Amazon S3 (p. 77).

Console Instructions

These steps are divided into three phases: initial channel setup, set up to auto-record to Amazon S3 (optional), and final channel creation.

Initial Channel Setup

1. Open the Amazon IVS console.
   (You can also access the Amazon IVS console through the AWS Management Console.)

2. From the navigation bar, use the Select a Region drop-down to choose a region. Your new channel will be created in this region.

3. In the Get started box (top right), choose Create Channel.

4. Under Channel configuration, accept the Default configuration. Optionally, specify a Channel name. Channel names are not unique, but they provide a way for you to distinguish channels other than the channel ARN (Amazon Resource Name).
5. If you want to auto-record to Amazon S3, continue with Set Up to Auto-Record to Amazon S3 (Optional) (p. 9) below. Otherwise, skip that and proceed directly to Final Channel Creation (p. 12).
Set Up to Auto-Record to Amazon S3 (Optional)

Follow these steps to enable recording while creating a new channel:

1. On the Create channel page, under Record and store streams, choose Auto-record to Amazon S3. Additional fields display, to choose an existing Recording configuration or create a new one.

2. Choose Create recording configuration. A new window opens, with Storage options for creating an Amazon S3 bucket and attaching it to the new recording configuration.
3. Leave the default selection as is, with **Create a new Amazon S3 bucket** selected. Enter a **Bucket Name**.
4. Choose **Create**, to create a new recording-configuration resource with a unique ARN. Typically, creation of the recording configuration takes a few seconds, but it can be up to 20 seconds. When the recording configuration is created, you are returned to the **Create channel** window. There, the **Record and store streams** area shows your new **Recording configuration**, with its **State** as **Active** and the **S3 Bucket** that you created.
Final Channel Creation

1. Choose **Create channel**, to create a new channel with a unique ARN. A channel details page is displayed for the new channel; keep this open. (Note: if you did not enable recording, **Auto-record to S3** is set to **Disabled** and there is no **Recording configuration** section on the screen.)
2. **Important:**
   - In the **Stream configuration** area, note the **Ingest server** and **Stream key**. You will use these in the next step, to set up streaming.
   - In the **Playback configuration** area, note the **Playback URL**. You will use it later, to play back your stream.
CLI Instructions

Creating a channel with the AWS CLI is an advanced option and requires that you first download and configure the CLI on your machine. For details, see the AWS Command Line Interface User Guide.

Follow one of the two procedures below, depending on whether you want to create a channel with or without recording enabled.

Create a Channel without Recording

1. Run the `create-channel` command and pass in an optional name:

   ```
   aws ivs create-channel --name test-channel
   ```

2. This returns a new channel:

   ```
   {
     "channel": {
       "authorized": false,
       "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",
       "latencyMode": "LOW",
       "name": "channel-live",
       "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/api/video/v1/us-west-2.123456789012.channel.abcdEFGH.m3u8",
       "recordingConfigurationArn": "none",
       "tags": {},
       "type": "STANDARD"
     },
     "streamKey": {
       "arn": "arn:aws:ivs:us-west-2:123456789012:stream-key/g1H2I3j4k5L6",
       "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
       "tags": {},
       "value": "sk_us-west-2_abcdABCDefgh_567890abcdef"
     }
   }
   ```

   Note: The `latencyMode` values in the API are LOW and NORMAL. In the Amazon IVS console, these correspond to Ultra-low and Standard, respectively.

3. Important: Note the `ingestEndpoint`, `streamKey` value, and `playbackUrl`. You will use these to set up streaming and playback.

Create a Channel with Recording

Prerequisite: Before starting this procedure, create an Amazon S3 bucket and note its ARN. See Getting Started with Amazon S3. The S3 bucket must be in the same region where you will create a recording configuration; see the known issue in Step 1 below.

Then follow these steps to create the channel:

1. Run the `create-recording-configuration` command and pass in the ARN of an existing Amazon S3 bucket:

   ```
   aws ivs create-recording-configuration --name configuration-1 --destination-configuration s3={bucketName=test-bucket}
   ```
Known issue: In the us-east-1 region, if you use the AWS CLI to create a recording configuration, it returns success even if the S3 bucket is in a different region. In this case, the state of the recording configuration is CREATE_FAILED (instead of ACTIVE). (In other regions, the CLI correctly returns failure if the bucket is in a different region.)

Workaround: Ensure that your S3 bucket is in the same region as the recording configuration. If you create a recording configuration in a different region as your S3 bucket, delete that recording configuration and create a new one with an S3 bucket from the correct region.

2. This returns a new recording configuration with a unique ARN. The state of the recording configuration is CREATING, indicating that it is in the process of being created.

```
{
  "recordingConfiguration": {
    "name": "configuration-1",
    "destinationConfiguration": {
      "s3": {
        "bucketName": "s3_bucket_name"
      }
    },
    "state": "CREATING",
    "tags": {}
  }
}
```

3. Typically, creation of the recording configuration takes a few seconds, but it can be up to 20 seconds. To check that the recording configuration has been created, run the get-recording-configuration command:

```
```

4. This returns a response indicating that the recording configuration was created (state is ACTIVE):

```
{
  "recordingConfiguration": {
    "name": "configuration-1",
    "destinationConfiguration": {
      "s3": {
        "bucketName": "s3_bucket_name"
      }
    },
    "state": "ACTIVE",
    "tags": {}
  }
}
```

5. To create a channel and enable recording on it, run the create-channel command and pass in the recording-configuration ARN:

```
aws ivs create-channel --name channel-live --recording-configuration-arn "arn:aws:ivs:us-west-2:123456789012:recording-configuration/mhndauNaO1te"
```

Alternately, to enable recording on an existing channel, run the update-channel command and pass in the recording-configuration ARN:
Step 4: Set Up Streaming Software

You can stream to Amazon IVS with the native mobile broadcast SDK or other streaming software (like the two examples below, Open Broadcast Software (OBS) and FFmpeg). These support RTMPS, like Amazon IVS. RTMPS enables high security via use of an encrypted TLS stream. You can use any streaming software that supports RTMPS.

Key encoder settings are keyframe interval (2 seconds) and resolution/bitrate/frame rate (which are interrelated). For more detail on encoder settings, see:

- Streaming Configuration (p. 115) in the Amazon IVS User Guide
- This blog post: Setting Up for Streaming with Amazon Interactive Video Service

Notes:

- The maximum duration of Amazon IVS streams is 48 hours. After that, the stream is terminated and the streaming session is disconnected. A successful reconnect (automatically or manually) starts a new stream.
- If your encoder stops sending data (e.g., due to a temporary network issue), Amazon IVS waits for 30 seconds. If no broadcaster data is received during this time, Amazon IVS disconnects.
Streaming with the Amazon IVS Broadcast SDK

To broadcast from your iOS or Android applications, you can use the Amazon IVS broadcast SDK. The broadcast SDK leverages the Amazon IVS architecture and will see continual improvement and new features, alongside Amazon IVS. As a native mobile broadcast SDK, it is designed to minimize the performance impact on your application and on the devices with which your users access your application.

For details, see Broadcasting to Amazon IVS (p. 20).

Streaming with OBS Studio

(OBS Studio) is a free, open-source software suite for recording and live streaming. OBS provides real-time source and device capture, scene composition, encoding, recording, and streaming.

Follow these steps to get up and running quickly with OBS Studio:

1. Download and install the software: https://obsproject.com/download.
2. Run the OBS Studio Auto-Configuration Wizard, which appears when you load OBS Studio for the first time. Follow the steps and accept the defaults.
3. When prompted, enter the Server and Stream Key.
   - If you created the channel with the Amazon IVS console:
     - The Server you enter in OBS is the Ingest server from the console, with the addition of port 443 (required for Amazon IVS ingest):
       ```
       rtmps://a1b2c3d4e5f6.global-contribute.live-video.net:443/app/
       ```
     - The Stream key you enter in OBS is the Stream key from the console:
       ```
       sk_us-west-2_abcd1234efgh5678ijkl
       ```
   - If you created the channel with the AWS CLI:
     - The Server you enter in OBS is a combination of four things:
       - An ingestion protocol: rtmps://
       - The ingestEndpoint from the CLI response:
         ```
         a1b2c3d4e5f6.global-contribute.live-video.net
         ```
       - A port: 443
       - A path: /app/
       - The complete entry is:
         ```
         rtmps://a1b2c3d4e5f6.global-contribute.live-video.net:443/app/
         ```
     - The Stream key you enter in OBS is the streamKey value from the CLI response:
       ```
       sk_us-west-2_abcd1234efgh5678ijkl
       ```
4. In the OBS Studio main window, choose Start Streaming.

For more on streaming with OBS Studio, see OBS Studio Quickstart.

You can modify your OBS settings manually later:

1. Choose Settings > Stream.
2. Choose Custom from the dropdown.
3. Paste in the **Server** and/or **Stream Key**.

You can run the wizard again at any time: choose **Tools > Auto-Configuration Wizard**.

Optionally, in **Settings > General**, enable local recording to save your live stream for later use. As mentioned earlier, network issues between the broadcast and AWS or within AWS could result in some data loss while recording your stream. In these cases, Amazon IVS prioritizes the live stream over the recording. Recording locally via your streaming tool provides redundancy.

It’s advisable to check for OBS Studio updates regularly and update to the most current version. (For instance, if you get a “Failed to connect to server” error, you may be using an old version of OBS Studio that does not support RTMPs.)

### Streaming a Recorded Video with FFmpeg

Follow these steps:

2. Set `$VIDEO_FILEPATH` to the location of an MP4 video to stream:

   ```
   VIDEO_FILEPATH=/home/test/my_video.mp4
   ```

3. Set `STREAM_KEY` to your StreamKey value:

   ```
   STREAM_KEY=sk_us-west-2_abcd1234efgh5678ijkl
   ```

4. Set `INGEST_ENDPOINT` to your **ingestEndpoint**:

   ```
   INGEST_ENDPOINT=a1b2c3d4e5f6.global-contribute.live-video.net
   ```

5. Start streaming with the following terminal command (this is all one line):

   ```
   ```

   *Note, the above command is an example. For production streaming, tune the parameters to your needs.*

### Step 5: View Your Live Stream

To view your live stream:

1. Open the **Amazon IVS console**.
2. In the top left, choose the hamburger icon to open the navigation pane, then choose **Live channels**.
3. Choose the channel whose stream you want to view, to go to a details page for that channel.

   The live stream is playing in the **Live stream** section of the page.

*Note: After you start streaming, there is a short delay (up to 30 seconds, usually less) before your stream can be viewed in the console.*
Step 6: Check Your Service-Quota Limits (Optional)

All accounts have limits on the number of concurrent viewers and concurrent broadcasts. Ensure that your limits are adequate and request an increase if needed, especially if you are planning a large streaming event. For details, see Amazon IVS Service Quotas (p. 111).

How to Disable Recording

To disable Amazon S3 recording on an existing channel:

- **Console** — On the details page for the relevant channel, in the Record and store streams section, choose Disabled and then choose Save Channel. This removes the recording configuration’s association with the channel; streams on that channel will no longer be recorded.
- **CLI** — Run the update-channel command and pass in the recording-configuration ARN as an empty string:

```
```

This returns a channel object with an empty string for recordingConfigurationArn, indicating that the recording is disabled.
Broadcasting to Amazon IVS

The Amazon Interactive Video Services (IVS) broadcast SDK is for developers who are building applications with Amazon IVS. This SDK is designed to leverage the Amazon IVS architecture and will see continual improvement and new features, alongside Amazon IVS. As a native broadcast SDK, it is designed to minimize the performance impact on your application and on the devices with which your users access your application.

Currently, the broadcast SDK is available only for Android and iOS applications.

Your application can leverage the key features of the Amazon IVS broadcast SDK:

- **High quality streaming** — The broadcast SDK supports high quality streaming. Capture video from your camera and encode it at up to 1080p quality for a high quality viewing experience.
- **Automatic Bitrate Adjustments** — Smartphone users are mobile, so their network conditions can change throughout the course of a broadcast. The Amazon IVS broadcast SDK automatically adjusts the video bitrate to accommodate changing network conditions.
- **Portrait and Landscape Support** — No matter how your users hold their devices, the image appears right-side up and properly scaled. The broadcast SDK supports both portrait and landscape canvas size. It automatically manages the aspect ratio when the users rotate their device away from the configured orientation.
- **Secure Streaming** — Your user's broadcasts are encrypted using TLS, so they can keep their streams secure.
- **External Audio Devices** — The Amazon IVS broadcast SDK supports audio jack, USB, and Bluetooth SCO external microphones.

## Platform Requirements

### Native Platforms

<table>
<thead>
<tr>
<th>Platform</th>
<th>Supported Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>5.0 (Lollipop) and later</td>
</tr>
<tr>
<td>iOS</td>
<td>iOS 11 and later</td>
</tr>
</tbody>
</table>

If broadcasting is essential to your application, specify Metal as a requirement for downloading your app from the Apple App Store, using `UIRequiredDeviceCapabilities`.

## Required Device Access

The broadcast SDK requires access to the device's cameras and microphones, both those built into the device and those connected through Bluetooth, USB, or audio jack.
Support

If you encounter a broadcast error or other issue with your stream, determine the unique playback session identifier via the broadcast API.

<table>
<thead>
<tr>
<th>For this Amazon IVS Broadcast SDK:</th>
<th>Use this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td><code>getSessionId</code> function</td>
</tr>
<tr>
<td>iOS</td>
<td><code>sessionId</code> property of <code>IVSBroadcastSession</code></td>
</tr>
<tr>
<td>Web</td>
<td><code>getSessionId</code> function</td>
</tr>
</tbody>
</table>

Share this broadcast session identifier with AWS support. With it, they can get information to help troubleshoot your issue.

**Note:** The broadcast SDK is continually improved. See Amazon IVS Release Notes (p. 142) for available versions and fixed issues. If appropriate, before contacting support, update your version of the broadcast SDK and see if that resolves your issue.

Versioning

The Amazon IVS broadcast SDKs use semantic versioning.

For this discussion, suppose:

- The latest release is 4.1.3.
- The latest release of the prior major version is 3.2.4.
- The latest release of version 1.x is 1.5.6.

Backward-compatible new features are added as minor releases of the latest version. In this case, the next set of new features will be added as version 4.2.0.

Backward-compatible, minor bug fixes are added as patch releases of the latest version. Here, the next set of minor bug fixes will be added as version 4.1.4.

Backward-compatible, major bug fixes are handled differently; these are added to several versions:

- Patch release of the latest version. Here, this is version 4.1.4.
- Patch release of the prior minor version. Here, this is version 3.2.5.
- Patch release of the latest version 1.x release. Here, this is version 1.5.7.

Major bug fixes are defined by the Amazon IVS product team. Typical examples are critical security updates and selected other fixes necessary for customers.

**Note:** In the examples above, released versions increment without skipping any numbers (e.g., from 4.1.3 to 4.1.4). In reality, one or more patch numbers may remain internal and not be released, so the released version could increment from 4.1.3 to, say, 4.1.6.

Also, version 1.x will be supported until the end of 2021 or when 3.x is released, whichever happens later.
Broadcasting to Amazon IVS: SDK for Android Guide

The Amazon Interactive Video (IVS) Android broadcast SDK provides the interfaces required to broadcast to IVS on Android.

The `com.amazonaws.ivs.broadcast` package implements the interface described in this document. The following operations are supported:

- Set up (initialize) a broadcast session.
- Manage broadcasting.
- Attach and detach input devices.
- Manage a composition session.
- Receive events.
- Receive errors.

**Latest version of Android broadcast SDK:** 1.0.0 ([Release Notes](https://aws.github.io/amazon-ivs-broadcast-docs/1.0.0/android/))

**Reference documentation:** For information on the most important methods available in the Amazon IVS Android broadcast SDK, see the reference documentation at https://aws.github.io/amazon-ivs-broadcast-docs/1.0.0/android/.

**Sample code:** See the Android sample repository on GitHub: https://github.com/aws-samples/amazon-ivs-broadcast-android-sample.

**Platform requirements:** Android 5.0 (Lollipop) or greater is required for development.

## Getting Started

### Install the Library

To add the Amazon IVS Android broadcast library to your Android development environment, add the library to your module's `build.gradle` file, as shown here (for the latest version of the Amazon IVS broadcast SDK):

```groovy
repositories {
    mavenCentral()
}

dependencies {
    implementation 'com.amazonaws:ivs-broadcast:1.0.0'
}
```

### Create the Event Listener

Setting up an event listener allows you to receive state updates, device-change notifications, errors, and session-audio information.

```java
BroadcastSession.Listener broadcastListener =
    new BroadcastSession.Listener() {
        @Override
        public void onStateChanged(@NonNull BroadcastSession.State state) {
            Log.d(TAG, "State" + state);
        }
    }
```
@Override
public void onError(@NonNull BroadcastException exception) {
    Log.e(TAG, "Exception: " + exception);
}
}

Request Permissions

Your app must request permission to access the user's camera and mic. (This is not specific to Amazon IVS; it is required for any application that needs access to cameras and microphones.)

Here, we check whether the user has already granted permissions and, if not, ask for them:

```java
final String[] requiredPermissions = {
    Manifest.permission.CAMERA, Manifest.permission.RECORD_AUDIO
};
for (String permission : requiredPermissions) {
    if (ContextCompat.checkSelfPermission(this, permission) != PackageManager.PERMISSION_GRANTED) {
        // If any permissions are missing we want to just request them all.
        ActivityCompat.requestPermissions(this, requiredPermissions, 0x100);
        break;
    }
}
```

Here, we get the user's response:

```java
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == 0x100) {
        for (int result : grantResults) {
            if (result == PackageManager.PERMISSION_DENIED) {
                return;
            }
        }
        setupBroadcastSession();
    }
}
```

Create the Broadcast Session

The broadcast interface is com.amazonaws.ivs.broadcast.BroadcastSession. Initialize it with a preset, as shown below. If there are any errors during initialization (such as a failure to configure a codec) your BroadcastListener will get an error message and broadcastSession.isReady will be false.

**Important:** All calls to the Amazon IVS Broadcast SDK for Android must be made on the thread on which the SDK is instantiated. A call from a different thread will cause the SDK to throw a fatal error and stop broadcasting.

```java
// Create a broadcast-session instance and sign up to receive broadcast events and errors.
Context ctx = getApplicationContext();
broadcastSession = new BroadcastSession(ctx,
    broadcastListener,
    Presets.Configuration.STANDARD_PORTRAIT,
    Presets.Devices.FRONT_CAMERA(ctx));
```
Also see Create the Broadcast Session (Advanced Version) (p. 25).

Set the ImagePreviewView for Preview

If you want to display a preview for an active camera device, add a preview `ImagePreviewView` for the device to your view hierarchy.

```java
// awaitDeviceChanges will fire on the main thread after all pending devices
// attachments have been completed
broadcastSession.awaitDeviceChanges(() -> {
    for(Device device: session.listAttachedDevices()) {
        // Find the camera we attached earlier
        if(device.getDescriptor().type == Device.Descriptor.DeviceType.CAMERA) {
            LinearLayout previewHolder = findViewById(R.id.previewHolder);
            ImagePreviewView preview = ((ImageDevice)device).getPreviewView();
            preview.setLayoutParams(new LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.MATCH_PARENT));
            previewHolder.addView(preview);
        }
    }
});
```

Start a Broadcast

The hostname that you receive in the `ingestEndpoint` response field of the `GetChannel` endpoint needs to have `rtmps://` prepended and `/app` appended. The complete URL should be in this format: `rtmps://{{ ingestEndpoint }}/app`

```java
broadcastSession.start(IVS_RTMP_URL, IVS_STREAMKEY);
```

Stop a Broadcast

```java
broadcastSession.stop();
```

Release the Broadcast Session

You **must call** the `broadcastSession.release()` method when the broadcast session is no longer in use, to free the resources used by the library.

```java
@Override
protected void onDestroy() {
    super.onDestroy();
    previewHolder.removeAllViews();
    broadcastSession.release();
}
```

Advanced Use Cases

Here we present some advanced use cases. Start with the basic setup above and continue here.

Create a Broadcast Configuration

Here we create a custom configuration with two mixer slots that allow us to bind two video sources to the mixer. One (`custom`) is full screen and laid out behind the other (`camera`), which is smaller and in
the bottom-right corner. Note that for the custom slot we do not set a position, size, or aspect mode. Because we do not set these parameters, the slot will use the video settings for size and position.

```java
BroadcastConfiguration config = BroadcastConfiguration.with($ -> {
    $.audio.setBitrate(128_000);
    $.video.setMaxBitrate(3_500_000);
    $.video.setMinBitrate(500_000);
    $.video.setInitialBitrate(1_500_000);
    $.video.setSize(1280, 720);
    $.mixer.slots = new BroadcastConfiguration.Mixer.Slot[] {
        BroadcastConfiguration.Mixer.Slot.with(slot -> {
            // Do not automatically bind to a source
            slot.setPreferredAudioInput(Device.Descriptor.DeviceType.UNKNOWN);
            // Bind to user image if unbound
            slot.setPreferredVideoInput(Device.Descriptor.DeviceType.USER_IMAGE);
            slot.setName("custom");
            return slot;
        }),
        BroadcastConfiguration.Mixer.Slot.with(slot -> {
            slot.setZIndex(1);
            slot.setAspect(BroadcastConfiguration.AspectMode.FILL);
            slot.setSize(300, 300);
            slot.setPosition($_.video.getSize().x - 200,
                            _.video.getSize().y - 200);
            slot.setName("camera");
            return slot;
        })
    });
    return $;
});
```

Create the Broadcast Session (Advanced Version)

Create a `BroadcastSession` as you did in the basic example (p. 23), but provide your custom configuration here. Also provide `null` for the device array, as we will add those manually.

```java
// Create a broadcast-session instance and sign up to receive broadcast
// events and errors.
Context ctx = getApplicationContext();
broadcastSession = new BroadcastSession(ctx,
broadcastListener,
    config, // The configuration we created above
    null); // We’ll manually attach devices after
```

Iterate and Attach a Camera Device

Here we iterate through input devices that the SDK has detected. On Android 7 (Nougat) this will only return default microphone devices, because the Amazon IVS Broadcast SDK does not support selecting non-default devices on this version of Android.

Once we find a device that we want to use, we call `attachDevice` to attach it. A lambda function is called on the main thread when attaching the input device has completed. In case of failure, you will receive an error in the Listener.

```java
for(Device.Descriptor desc: BroadcastSession.listAvailableDevices(getApplicationContext()))
{
    if(desc.type == Device.Descriptor.DeviceType.CAMERA &&
```
 desc.position == Device.Descriptor.Position.FRONT) {
    session.attachDevice(desc, device -> {
        LinearLayout previewHolder = findViewById(R.id.previewHolder);
        ImagePreviewView preview = ((ImageDevice)device).getPreviewView();
        preview.setLayoutParams(new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.MATCH_PARENT));
        previewHolder.addView(preview);
        // Bind the camera to the mixer slot we created above.
        session.getMixer().bind(device, "camera");
    });
    break;
}

Swap Cameras

// This assumes you’ve kept a reference called "currentCamera" that points to
// a front facing camera
for(Device device: BroadcastSession.listAvailableDevices()) {
    if(device.type == Device.Descriptor.DeviceType.CAMERA &&
       Device.position != currentCamera.position) {
        // Remove the preview view for the old device.
        // setImagePreviewTextureView is an example function
        // that handles your view hierarchy.
        session.exchangeDevices(currentCamera, device, camera -> {
            // Set the preview view for the new device.
            session.exchangeDevices(currentCamera, device, camera -> {
                setImagePreviewView(null);
                // Set the preview view for the new device.
                session.exchangeDevices(currentCamera, device, camera -> {
                    setImagePreviewView(camera.getPreviewView());
                    currentCamera = camera;
                });
                break;
            });
            break;
        });
        break;
    }
}

Create an Input Surface

To input sound or image data that your app generates, use createImageInputSource or
createAudioInputSource. Both these methods create and attach virtual devices that can be bound to
the mixer like any other device.

The SurfaceSource returned by createImageInputSource has a getInputSurface method, which
will give you a Surface that you can use with the Camera2 API, OpenGL, or Vulkan, or anything else
that can write to a Surface.

The AudioDevice returned by createAudioInputSource can receive Linear PCM data generated by
AudioRecorder or other means.

SurfaceSource source = session.createImageInputSource();
Surface surface = source.getInputSurface();
session.getMixer().bind(source, "custom");

Detach a Device

If you want to detach and not replace a device, detach it with Device or Device.Descriptor.

session.detachDevice(currentCamera);
Screen and System Audio Capture

The Amazon IVS Broadcast SDK for Android includes some helpers that simplify capturing the device's screen (Android 5 and higher) and system audio (Android 10 and higher). If you want to manage these manually, you can create a custom image-input source and a custom audio-input source.

To create a screen and system audio-capture session, you must first create a permission-request intent:

```java
public void startScreenCapture() {
    MediaProjectionManager manager =
        (MediaProjectionManager) getApplicationContext()
            .getSystemService(Context.MEDIA_PROJECTION_SERVICE);
    if(manager != null) {
        Intent intent = manager.createScreenCaptureIntent();
        startActivityIf Needed(intent, SCREEN_CAPTURE_REQUEST_ID);
    }
}
```

To use this feature, you must provide a class that extends `com.amazonaws.ivs.broadcast.SystemCaptureService`. You do not have to override any of its methods, but the class needs to be there to avoid any potential collisions between services.

You also must add a couple of elements to your Android manifest:

```xml
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
<application ...>
    <service android:name=".ExampleSystemCaptureService"
            android:foregroundServiceType="mediaProjection"
            android:isolatedProcess="false" />
</application>
```

Your class that extends `SystemCaptureService` must be named in the `<service>` element. On Android 9 and later, the `foregroundServiceType` must be `mediaProjection`.

Once the permissions intent has returned, you may proceed with creating the screen and system audio-capture session. On Android 8 and later, you must provide a notification to be displayed in your user's Notification Panel. The Amazon IVS Broadcast SDK for Android provides the convenience method `createServiceNotificationBuilder`. Alternately, you may provide your own notification.

```java
@override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode != SCREEN_CAPTURE_REQUEST_ID
        || Activity.RESULT_OK != resultCode) {
        return;
    }
    Notification notification = null;
    if(Build.VERSION.SDK_INT >= 26) {
        Intent intent = new Intent(getApplicationContext(),
            NotificationActivity.class);
        notification = session
            .createServiceNotificationBuilder("example",
            "example channel", intent)
            .build();
    }
    session.createSystemCaptureSources(data,
        ExampleSystemCaptureService.class,
        Notification,
        devices -> { // This step is optional if the mixer slots have been given preferred
```
Get Recommended Broadcast Settings

To evaluate your user's connection before starting a broadcast, use the `recommendedVideoSettings` method to run a brief test. As the test runs, you will receive several recommendations, ordered from most to least recommended. In this version of the SDK, it is not possible to reconfigure the current `BroadcastSession`, so you will need to `release()` it and then create a new one with the recommended settings. You will continue to receive `BroadcastSessionTest.Results` until the `Result.status` is `SUCCESS` or `ERROR`. You can check progress with `Result.progress`.

Amazon IVS supports a maximum bitrate of 8.5 Mbps (for channels whose `type` is `STANDARD`), so the `maximumBitrate` returned by this method never exceeds 8.5 Mbps. To account for small fluctuations in network performance, the recommended `initialBitrate` returned by this method is slightly less than the true bitrate measured in the test. (Using 100% of the available bandwidth usually is inadvisable.)

```
void runBroadcastTest() {
    this.test = session.recommendedVideoSettings(RTMP_ENDPOINT, RTMP_STREAMKEY, 
        result -> {
            if (result.status == BroadcastSessionTest.Status.SUCCESS) {
                this.recommendation = result.recommendations[0];
            }
        });
}
```

Known Issues and Workarounds

Please report all issues to Amazon IVS Support.

- Using an external microphone connected through Bluetooth can be unstable. When a Bluetooth device is connected or disconnected during a broadcasting session, microphone input may stop working until the device is explicitly detached and reattached.
  
  **Workaround:** If you plan to use a Bluetooth headset, connect it before starting the broadcast and leave it connected throughout the broadcast.

- The broadcast SDK does not support access on external cameras connected via USB.
  
  **Workaround:** Do not use external cameras connected via USB.

- Submitting audio data faster than realtime (using a custom audio source) results in audio drift.
  
  **Workaround:** Do not submit audio data faster than realtime.

- Some Android 5 devices may stream a black image if the same `BroadcastSession` is used for multiple broadcasts.
  
  **Workaround:** When stopping the `BroadcastSession`, release it and instantiate a new one.

Broadcasting to Amazon IVS: SDK for iOS Guide

The Amazon Interactive Video Service (IVS) iOS broadcast SDK provides the interfaces required to broadcast to Amazon IVS on iOS.
The AmazonIVSBroadcast module implements the interface described in this document. The following operations are supported:

- Set up (initialize) a broadcast session.
- Manage broadcasting.
- Attach and detach input devices.
- Manage a composition session.
- Receive events.
- Receive errors.

**Latest version of iOS broadcast SDK: 1.0.0 (Release Notes)**

**Reference documentation:** For information on the most important methods available in the Amazon IVS iOS broadcast SDK, see the reference documentation at [https://aws.github.io/amazon-ivs-broadcast-docs/1.0.0/ios/](https://aws.github.io/amazon-ivs-broadcast-docs/1.0.0/ios/).

**Sample code:** See the iOS sample repository on GitHub: [https://github.com/aws-samples/amazon-ivs-broadcast-ios-sample](https://github.com/aws-samples/amazon-ivs-broadcast-ios-sample).

**Platform requirements:** iOS 11 or greater.

## Getting Started

### Install the Library

We recommend that you integrate the player SDK via CocoaPods. (Alternatively, you can manually add the framework to your project.)

**Recommended: Integrate the Broadcast SDK (CocoaPods)**

Releases are published via CocoaPods under the name AmazonIVSBroadcast. Add this dependency to your Podfile:

```
pod 'AmazonIVSBroadcast'
```

Run `pod install` and the SDK will be available in your `.xcworkspace`.

**Alternate Approach: Install the Framework Manually**

1. Download the latest version from [https://broadcast.live-video.net/1.0.0/AmazonIVSBroadcast.xcframework.zip](https://broadcast.live-video.net/1.0.0/AmazonIVSBroadcast.xcframework.zip).
2. Extract the contents of the archive. AmazonIVSBroadcast.xcframework contains the SDK for both device and simulator.
3. Embed AmazonIVSBroadcast.xcframework by dragging it into the **Frameworks, Libraries, and Embedded Content** section of the **General** tab for your application target.
Implement IVSBroadcastSession.Delegate

Implement `IVSBroadcastSession.Delegate`, which allows you to receive state updates and device-change notifications:

```swift
extension ViewController : IVSBroadcastSession.Delegate {
    func broadcastSession(_ session: IVSBroadcastSession, didChange state: IVSBroadcastSession.State) {
        print("IVSBroadcastSession did change state \(state)")
    }

    func broadcastSession(_ session: IVSBroadcastSession, didEmitError error: Error) {
        print("IVSBroadcastSession did emit error \(error)")
    }
}
```

Request Permissions

Your app must request permission to access the user’s camera and mic. (This is not specific to Amazon IVS; it is required for any application that needs access to cameras and microphones.)

Here, we check whether the user has already granted permissions and, if not, we ask for them:

```swift
switch AVCaptureDevice.authorizationStatus(for: .video) {
    case .authorized: // permission already granted.
    case .notDetermined:
        AVCaptureDevice.requestAccess(for: .video) { granted in
            // permission granted based on granted bool.
        }
    case .denied, .restricted: // permission denied.
    @unknown default: // permissions unknown.
}
```

You need to do this for both `.video` and `.audio` media types, if you want access to cameras and microphones, respectively.

You also need to add entries for `NSCameraUsageDescription` and `NSMicrophoneUsageDescription` to your `Info.plist`. Otherwise, your app will crash when trying to request permissions.

Disable the Application Idle Timer

This is optional but recommended. It prevents your device from going to sleep while using the broadcast SDK, which would interrupt the broadcast.

```swift
override func viewDidAppear(_ animated: Bool) {
    super.viewDidAppear(animated)
    UIApplication.shared.isIdleTimerDisabled = true
}

override func viewDidDisappear(_ animated: Bool) {
    super.viewDidDisappear(animated)
    UIApplication.shared.isIdleTimerDisabled = false
}
```

(Optional) Set Up AVAudioSession

By default, the broadcast SDK will set up your application’s `AVAudioSession`. If you want to manage this yourself, set this to `noAction`: `IVSBroadcastSession.applicationAudioSessionStrategy`
If you are manually setting up your AVAudioSession, at a minimum you need to set the category as .record or .playbackAndRecord, and set it to active. If you want to record audio from Bluetooth devices, you need to specify the .allowBluetooth option as well:

```swift
do {
    try AVAudioSession.sharedInstance().setCategory(.record, options: .allowBluetooth)
    try AVAudioSession.sharedInstance().setActive(true)
} catch {
    print("Error configuring AVAudioSession")
}
```

We recommend that you let the SDK handle this for you. Otherwise, if you want to choose between different audio devices, you will need to manually manage the ports.

### Create the Broadcast Session

The broadcast interface is IVSBroadcastSession. Initialize it as shown below:

```swift
let broadcastSession = try IVSBroadcastSession(
    configuration: IVSPresets.configurations().standardLandscape(),
    descriptors: IVSPresets.devices().frontCamera(),
    delegate: self)
```

Also see Create the Broadcast Session (Advanced Version) (p. 34)

### Set the IVSImagePreviewView for Preview

If you want to display a preview for an active camera device, add the preview IVSImagePreviewView for the device to your view hierarchy:

```swift
// If the session was just created, execute the following
// code in the callback of IVSBroadcastSession.awaitDeviceChanges
// to ensure all devices have been attached.
if let devicePreview = try broadcastSession.listAttachedDevices().compactMap({ $0 as? IVSImageDevice }).first?.previewView() {
    previewView.addSubview(devicePreview)
}
```

### Start a Broadcast

The hostname that you receive in the ingestEndpoint response field of the GetChannel endpoint needs to have rtmps:// prepended and /app appended. The complete URL should be in this format: rtmps://{{ ingestEndpoint }}/app.

```swift
try broadcastSession.start(with: IVS_RTMP_URL, streamKey: IVS_STREAMKEY)
```

### Stop a Broadcast

```swift
broadcastSession.stop()
```
Manage Lifecycle Events

Audio Interruptions

There are several scenarios where the broadcast SDK will not have exclusive access to audio-input hardware. Some example scenarios that you need to handle are:

- User receives a phone call or FaceTime call
- User activates Siri

Apple makes it easy to respond to these events by subscribing to AVAudioSession.interruptionNotification:

```swift
NotificationCenter.default.addObserver(
    self,
    selector: #selector(audioSessionInterrupted(_:)),
    name: AVAudioSession.interruptionNotification,
    object: nil)
```

Then you can handle the event with something like this:

```swift
// This assumes you have a variable `isRunning` which tracks if the broadcast is currently live, and another variable `wasRunningBefore Interruption` which tracks whether the broadcast was active before this interruption to determine if it should resume after the interruption has ended.

@objc
private func audioSessionInterrupted(_ notification: Notification) {
    guard let userInfo = notification.userInfo,
         let typeValue = userInfo[AVAudioSessionInterruptionTypeKey] as? UInt,
         let type = AVAudioSession.InterruptionType(rawValue: typeValue)
    else {
        return
    }

    switch type {
    case .began:
        wasRunningBeforeInteruption = isRunning
        if isRunning {
            broadcastSession.stop()
        }
    case .ended:
        defer {
            wasRunningBeforeInteruption = false
        }
        guard let optionsValue = userInfo[AVAudioSessionInterruptionOptionKey] as? UInt else {
            return
        }
        let options = AVAudioSession.InterruptionOptions(rawValue: optionsValue)
        if options.contains(.shouldResume) && wasRunningBeforeInteruption {
            try broadcastSession.start(
                with: IVS_RTMPs_URL,
                streamKey: IVS_STREAMKEY)
        }
    case .unknown: break
    }
}
```

App Going Into Background

Standard applications on iOS are not allowed to use cameras in the background. There also are restrictions on video encoding in the background: since hardware encoders are limited, only foreground
applications have access. Because of this, the broadcast SDK automatically terminates its session and sets its `isReady` property to `false`. When your application is about to enter the foreground again, the broadcast SDK reattaches all the devices to their original `IVSMixerSlotConfiguration` entries.

The broadcast SDK does this by responding to `UIApplication.didEnterBackgroundNotification` and `UIApplication.willEnterForegroundNotification`.

If you are providing custom image sources, you should be prepared to handle these notifications. You may need to take extra steps to tear them down before the stream is terminated.

**Media Services Lost**

In very rare cases, the entire media subsystem on an iOS device will crash. In this scenario, we can no longer broadcast. It is up to your application to respond to these notifications appropriately. At a minimum, subscribe to these notifications:

- `mediaServicesWereLostNotification` — Respond by stopping your broadcast and completely deallocating your `IVSBroadcastSession`. All internal components used by the broadcast session will be invalidated.
- `mediaServicesWereResetNotification` — Respond by notifying your users that they can broadcast again. Depending on your use case, you may be able to automatically start broadcasting again at this point.

**Advanced Use Cases**

Here we present some advanced use cases. Start with the basic setup above and continue here.

**Create a Broadcast Configuration**

Here we create a custom configuration with two mixer slots that allow us to bind two video sources to the mixer. One (`custom`) is full screen and laid out behind the other (`camera`), which is smaller and in the bottom-right corner. Note that for the `custom` slot we do not set a position, size, or aspect mode. Because we do not set these parameters, the slot uses the video settings for size and position.

```swift
let config = IVSBroadcastConfiguration()
try config.audio.setBitrate(128_000)
try config.video.setMaxBitrate(3_500_000)
try config.video.setMinBitrate(500_000)
try config.video.setInitialBitrate(1_500_000)
try config.video.setSize(CGSize(width: 1280, height: 720))
config.video.defaultAspectMode = .fit
config.mixer.slots = [
    try {
        let slot = IVSMixerSlotConfiguration()
        // Do not automatically bind to a source
        slot.preferredAudioInput = .unknown
        // Bind to user image if unbound
        slot.preferredVideoInput = .userImage
        try slot.setName("custom")
        return slot
    }(),
    try {
        let slot = IVSMixerSlotConfiguration()
        slot.zIndex = 1
        slot.aspect = .fill
        slot.size = CGSize(width: 300, height: 300)
        slot.position = CGPoint(x: config.video.size.width - 200, y: config.video.size.height - 200)
        try slot.setName("camera")
        return slot
    }
]
```
Create the Broadcast Session (Advanced Version)

Create an IVSBroadcastSession as you did in the basic example (p. 31), but provide your custom configuration here. Also provide nil for the device array, as we will add those manually.

```swift
let broadcastSession = try IVSBroadcastSession(
    configuration: config, // The configuration we created above
    descriptors: nil, // We’ll manually attach devices after
    delegate: self)
```

Iterate and Attach a Camera Device

Here we iterate through input devices that the SDK has detected. The SDK will only return built-in devices on iOS. Even if Bluetooth audio devices are connected, they will appear as a built-in device. For more information, see Known Issues and Workarounds (p. 36).

Once we find a device that we want to use, we call `attachDevice` to attach it:

```swift
let frontCamera = IVSBroadcastSession.listAvailableDevices()
    .filter { $0.type == .camera && $0.position == .front }
    .first
if let camera = frontCamera {
    broadcastSession.attach(camera, toSlotWithName: "camera") { device, error in
        // check error
    }
}
```

Swap Cameras

// This assumes you’ve kept a reference called `currentCamera` that points to the current camera.
let wants: IVSDevicePosition = (currentCamera.descriptor().position
  == .front) ? .back : .front
// Remove the current preview view since the device will be changing.
previewView.subviews.forEach { $0.removeFromSuperview() }
let foundCamera = IVSBroadcastSession
    .listAvailableDevices()
    .first { $0.type == .camera && $0.position == wants }
guard let newCamera = foundCamera else { return }
broadcastSession.exchangeOldDevice(currentCamera, withNewDevice: newCamera) { newDevice, _
    in
    currentCamera = newDevice
    if let camera = newDevice as? IVSImageDevice {
        do {
            previewView.addSubview(try finalCamera.previewView())
        }
        catch {
            print("Error creating preview view \(error)")
        }
    }
}
```

Create a Custom Input Source

To input sound or image data that your app generates, use `createImageSource` or `createAudioSource`. Both these methods create virtual devices that can be bound to the mixer like any other device.
The devices returned by both these methods accept a CMSampleBuffer through its onSampleBuffer function:

- For video sources, the pixel format must be kCVPixelFormatType_32BGRA, 420YpCbCr8BiPlanarFullRange, or 420YpCbCr8BiPlanarVideoRange.
- For audio sources, the buffer must contain Linear PCM data.

You cannot use an AVCaptureSession with camera input to feed a custom image source while also using a camera device provided by the broadcast SDK. If you want to use multiple cameras simultaneously, use AVCaptureMultiCamSession and provide two custom image sources.

Custom image sources primarily should be used with static content such as images, or with video content:

```swift
let customImageSource = broadcastSession.createImageSource(withName: "video")
try broadcastSession.attach(customImageSource, toSlotWithName: "custom")
```

**Monitor Network Connectivity**

It is common for mobile devices to temporarily lose and regain network connectivity while on the go. Because of this, it is important to monitor your app's network connectivity and respond appropriately when things change.

When the broadcaster's connection is lost, the broadcast SDK's state will change to error and then disconnected. You will be notified of these changes through the IVSBroadcastSessionDelegate. When you receive these state changes:

1. Monitor your broadcast app's connectivity state and call start with your endpoint and stream key, once your connection has been restored.
2. **Important:** Monitor the state delegate callback and ensure that the state changes to connected after calling start again.

**Detach a Device**

If you want to detach and not replace a device, detach it with IVSDevice or IVSDeviceDescriptor:

```swift
broadcastSession.detachDevice(currentCamera)
```

**ReplayKit Integration**

To stream the device's screen and system audio on iOS, you must integrate with ReplayKit. The Amazon IVS Player SDK makes it easy to integrate ReplayKit using IVSReplayKitBroadcastSession. In your RPBroadcastSampleHandler subclass, create an instance of IVSReplayKitBroadcastSession, then:

- Start the session in broadcastStarted
- Stop the session in broadcastFinished

The session object will have three custom sources for screen images, app audio, and microphone audio. Pass the CMSampleBuffers provided in processSampleBuffer to those custom sources.

To handle device orientation, you need to extract ReplayKit-specific metadata from the sample buffer. Use the following code:
let imageSource = session.systemImageSource;
if #available(iOSApplicationExtension 11.0, *) {
    if let orientationAttachment = CMGetAttachment(sampleBuffer, key: RPVideoSampleOrientationKey as CFString, attachmentModeOut: nil) as? NSNumber,
    let orientation = CGImagePropertyOrientation(rawValue: orientationAttachment.uint32Value) {
        switch orientation {
        case .up, .upMirrored:
            imageSource.setHandsetRotation(0)
        case .down, .downMirrored:
            imageSource.setHandsetRotation(Float.pi)
        case .right, .rightMirrored:
            imageSource.setHandsetRotation(-(Float.pi / 2))
        case .left, .leftMirrored:
            imageSource.setHandsetRotation((Float.pi / 2))
        }
    }
}

It is possible to integrate ReplayKit using IVSBroadcastSession instead of IVSReplayKitBroadcastSession. However, the ReplayKit-specific variant has several modifications to reduce the internal memory footprint, to stay within Apple’s memory ceiling for broadcast extensions.

**Get Recommended Broadcast Settings**

To evaluate your user’s connection before starting a broadcast, use the recommendedVideoSettings method to run a brief test. As the test runs, you will receive several recommendations, ordered from most to least recommended. In this version of the SDK, it is not possible to reconfigure the current IVSBroadcastSession, so you must deallocatate it and then create a new one with the recommended settings. You will continue to receive IVSBroadcastSessionTestResults until the result.status is Success or Error. You can check progress with result.progress.

Amazon IVS supports a maximum bitrate of 8.5 Mbps (for channels whose type is STANDARD), so the maximumBitrate returned by this method never exceeds 8.5 Mbps. To account for small fluctuations in network performance, the recommended initialBitrate returned by this method is slightly less than the true bitrate measured in the test. (Using 100% of the available bandwidth usually is inadvisable.)

```swift
func runBroadcastTest() {
    self.test = session.recommendedVideoSettings(with: IVS_RTMPS_URL, streamKey: IVS_STREAMKEY) {
        [weak self] result in
        if result.status == .success {
            this.recommendation = result.recommendations[0];
        }
    }
}
```

**Known Issues and Workarounds**

Please report all issues to Amazon IVS Support.

- Changing Bluetooth audio routes can be unpredictable. If you connect a new device mid-session, iOS may or may not automatically change the input route. Also, it is not possible to choose between multiple Bluetooth headsets that are connected at the same time.

  **Workaround:** If you plan to use a Bluetooth headset, connect it before starting the broadcast and leave it connected throughout the broadcast.

- A bug in ReplayKit causes rapid memory growth when plugging in a wired headset during a stream.
**Workaround:** Start the stream with the wired headset already plugged in, use a Bluetooth headset, or do not use an external microphone.

- If at any point during a ReplayKit stream you enable the microphone and then interrupt the audio session (e.g., with a phone call or by activating Siri), system audio will stop working. This is a ReplayKit bug that we are working with Apple to resolve.

  **Workaround:** On an audio interruption, stop the broadcast and alert the user.

- **AirPods** do not record any audio if the `AVAudioSession` category is set to `record`. By default, the SDK uses `playAndRecord`, so this issue manifests only if the category is changed to `record`.

  **Workaround:** If there is a chance that AirPods will be used to record audio, use `playAndRecord` even if your application is not playing back media.

- Submitting audio data faster than realtime (using a custom audio source) results in audio drift.

  **Workaround:** Do not submit audio data faster than realtime.
Amazon Interactive Video Service Player

To use Amazon Interactive Video Service (IVS), you must use the Amazon IVS Player. The Player is a cross-platform suite of SDKs for playback of Amazon IVS streams. It is designed to leverage the Amazon IVS architecture and optimized for Amazon IVS playback.

The only player whose performance we can guarantee is the Amazon IVS player. To achieve low latency, the Amazon IVS player is required.

Key features of the Amazon IVS player are:

- **Ultra-low-latency streaming** — Low latency is a critical component in building good interactive user experiences that enrich the audience experience. Latency creeps in incrementally throughout the transmission path between broadcaster and viewer, eroding responsiveness.

  End-to-end latency is the delay from when a live stream is captured on camera to when it appears on a viewer's screen. Amazon IVS is designed to deliver ultra-low end-to-end latency (under five seconds, depending on the broadcast location and broadcaster settings). To achieve this low latency, the Amazon IVS player is required.

- **Cross-platform consistency** — Viewers watch broadcasts on a variety of platforms. From mobile devices to web browsers, the Amazon IVS Player gives all viewers a similar experience. This consistency is possible because every platform uses the same library of player functions. The player library is an integral component of the Amazon IVS architecture. Using one video stack ensures that all video-playback behaviors — including low-latency mode, timed metadata, analytics, error tracking, reporting, and logging — are available in a consistent way on all supported platforms.

- **Adaptive bitrate streaming (ABR)** — The Amazon IVS Player uses ABR algorithms optimized for low-latency environments. The Player measures quality of service and bandwidth availability in real time and adapts video quality and buffer levels, to provide uninterrupted playback. When connection quality suffers, ABR switches to a lower bitrate; when connection quality improves, it switches to a higher bitrate.

- **Timed metadata** — The Amazon IVS Player supports timed metadata, which can be used to build interactive elements such as polls and quizzes. Metadata is a set of data that describes and gives information about other data. With “timed” metadata, a timecode accompanies the piece of data about the stream. During playback, the timecode serves as a cue point to trigger action based on the data, such as:
  - Sending player statistics for a sports stream
  - Sending product details for a live shopping stream
  - Sending questions for a live quiz stream

- **Robust error handling** — Handling transient errors well avoids interruptions in the viewing experience. The Amazon IVS Player’s robust error handling detects many potential streaming errors, automatically switching to an alternative rendition. Viewers continue watching the broadcast uninterrupted, without having to take any corrective action.

- **Ease of integration** — The Amazon IVS Player API bridges the gap between Amazon IVS customers’ applications and the Player library. The API has bindings for all supported platforms, making it easy to integrate the Player into applications while using familiar coding environments and techniques. With full control over UI elements, customers can customize the branding and presentation aspects of their applications.
Browser & Platform Requirements

For details on the latest released versions of various browsers, see:

- Chrome Platform Status
- Firefox Releases
- Microsoft Edge Release Schedule
- Safari Release Notes

While Amazon IVS may work with some older browsers, we do not fix bugs related to older browsers.

**Desktop Browsers**

<table>
<thead>
<tr>
<th>Desktop Browser</th>
<th>Supported Platforms</th>
<th>Supported Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome</td>
<td>Windows, macOS</td>
<td>Two major versions (current and most recent prior version)</td>
</tr>
<tr>
<td>Firefox</td>
<td>Windows, macOS</td>
<td>Two major versions (current and most recent prior version)</td>
</tr>
<tr>
<td>Edge</td>
<td>Windows 8.1 and later</td>
<td>44.0 and later</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(In auto-quality mode on Microsoft Edge Legacy, only normal-latency playback is supported, not ultra-low latency playback. Auto-quality mode refers to whether ABR is enabled. For example, on the Web player, see <code>setAutoQualityMode</code>.</td>
</tr>
<tr>
<td>Safari</td>
<td>macOS</td>
<td>Two major versions (current and most recent prior version)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(In auto-quality mode on Safari for macOS 14 and above, IVS Player 1.3.0 and above support ultra-low latency playback. For earlier versions of Safari and IVS Player, only normal-latency playback is supported. See above for &quot;auto-quality mode.&quot; )</td>
</tr>
</tbody>
</table>

**Mobile Browsers**

<table>
<thead>
<tr>
<th>Mobile Browser</th>
<th>Supported Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome for iOS, Safari for iOS</td>
<td>Two major versions (current and most recent prior version)</td>
</tr>
<tr>
<td></td>
<td>(Ultra-low latency playback is not supported. Normal latency playback is supported. This constraint applies to all browsers for iOS.)</td>
</tr>
<tr>
<td></td>
<td>(Timed metadata is supported only in Player 1.3.0 and later.)</td>
</tr>
<tr>
<td>Chrome for iPadOS, Safari for iPadOS</td>
<td>Two major versions (current and most recent prior version)</td>
</tr>
<tr>
<td></td>
<td>(When &quot;Request Mobile Website&quot; is selected:</td>
</tr>
</tbody>
</table>
Native Platforms

<table>
<thead>
<tr>
<th>Platform</th>
<th>Supported Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>5.0 (Lollipop) and later</td>
</tr>
<tr>
<td>iOS</td>
<td>10.0 and later</td>
</tr>
</tbody>
</table>

Support

If you encounter a playback error or other playback issue with your stream, determine the unique playback session identifier via the player API.

<table>
<thead>
<tr>
<th>For this Amazon IVS player:</th>
<th>Use this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>sessionId function</td>
</tr>
<tr>
<td>iOS</td>
<td>sessionId property of IVSPlayer</td>
</tr>
<tr>
<td>Web</td>
<td>getSessionId function</td>
</tr>
</tbody>
</table>

Share this playback session identifier with AWS support. With it, they can get information to help troubleshoot your issue.

Note: The Player is continually improved. See Amazon IVS Release Notes (p. 142) for available versions and fixed issues. If appropriate, before contacting support, update your version of the Player and see if that resolves your issue.

Versioning

The Amazon IVS Player SDKs use semantic versioning.

For this discussion, suppose:

- The latest release is 4.1.3.
- The latest release of the prior major version is 3.2.4.
- The latest release of version 1.x is 1.5.6.

Backward-compatible new features are added as minor releases of the latest version. In this case, the next set of new features will be added as version 4.2.0.

Backward-compatible, minor bug fixes are added as patch releases of the latest version. Here, the next set of minor bug fixes will be added as version 4.1.4.
Backward-compatible, major bug fixes are handled differently; these are added to several versions:

- Patch release of the latest version. Here, this is version 4.1.4.
- Patch release of the prior minor version. Here, this is version 3.2.5.
- Patch release of the latest version 1.x release. Here, this is version 1.5.7.

Major bug fixes are defined by the Amazon IVS product team. Typical examples are critical security updates and selected other fixes necessary for customers.

Note: In the examples above, released versions increment without skipping any numbers (e.g., from 4.1.3 to 4.1.4). In reality, one or more patch numbers may remain internal and not be released, so the released version could increment from 4.1.3 to, say, 4.1.6.

Also, version 1.x will be supported until the end of 2021 or when 3.x is released, whichever happens later.

Amazon Interactive Video Service Player: SDK for Web Guide

The Amazon Interactive Video Service (IVS) Player SDK for Web can be integrated with player frameworks (p. 42) like Video.js or used standalone on top of an HTML `<video>` element.

Latest version of Web player: 1.4.1 (Release Notes)

Reference documentation: For information on the most important methods available in the Amazon IVS Web player, see the reference documentation at https://aws.github.io/amazon-ivs-player-docs/1.4.1/web/.

Getting Started

We provide support through a script tag as well as through an npm module.

Demo

The following live demo shows how to use the Web player with a script tag from our Content Delivery Network: Amazon IVS Player Sample.

Setup With Script Tag

To set up the Amazon IVS player using the script tag:

1. Include the following tag (for the latest version of the player).

   `<script src="https://player.live-video.net/1.4.1/amazon-ivs-player.min.js">`

2. Once `amazon-ivs-player.min.js` is loaded, it adds an IVSPlayer variable to the global context. This is the library you will use to create a player instance. First, check `isPlayerSupported` to determine if the browser supports the IVS player:

   ```javascript
   if (IVSPlayer.isPlayerSupported) { ... }
   ```
Then, to create a player instance, call the `create` function on the `IVSPlayer` object.

```javascript
const player = IVSPlayer.create();
```

The Amazon IVS Player SDK for Web uses web workers to optimize video playback.

3. Load and play a stream using the `load` and `play` functions on the player instance:

```javascript
player.load(PLAYBACK_URL);
player.play();
```

where `PLAYBACK_URL` is the URL returned from the Amazon IVS API when a stream key is requested.

**Sample Code**

In this example, `PLAYBACK_URL` is the source stream you want to load. The example uses the latest version of the Amazon IVS player.

```html
<script src="https://player.live-video.net/1.4.1/amazon-ivs-player.min.js"></script>
<video id="video-player" playsinline></video>
<script>
if (IVSPlayer.isPlayerSupported) {
  const player = IVSPlayer.create();
  player.attachHTMLVideoElement(document.getElementById('video-player'));
  player.load(PLAYBACK_URL);
  player.play();
}
</script>
```

In the `<video>` tag, `playsinline` is required for inline playback on iOS Safari. See [https://webkit.org/blog/6784/new-video-policies-for-ios/](https://webkit.org/blog/6784/new-video-policies-for-ios/).

**Setup With NPM**

For guidance, including an example Webpack configuration file, see the following repository: [https://github.com/aws-samples/amazon-ivs-player-web-sample](https://github.com/aws-samples/amazon-ivs-player-web-sample).

**Note:** When hosting player static assets from your own domain, you must set the "Content-Type" response header for the WebAssembly binary (`amazon-ivs-wasmworker.min.wasm`) to "application/wasm." You also should gzip your assets to reduce bytes downloaded over the wire and improve the player's time to start playback.

**TypeScript**

If you're using TypeScript, the npm package includes types you may want to import and use. For information on these types, see the Amazon IVS Player: SDK for Web Reference.

**Framework Integrations**

The Amazon IVS Player SDK for Web is designed to be easy to integrate with your framework of choice. We offer an official Video.js integration ("tech," in Video.js jargon).

The following is a brief comparison of the Web players we offer:
### Player Type

<table>
<thead>
<tr>
<th>Player Type</th>
<th>Description</th>
<th>UI</th>
<th>Plugins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon IVS Player SDK for Web</td>
<td>A lightweight and customizable option for developers who want more control.</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Amazon IVS Player Tech for Video.js (p. 53)</td>
<td>A full-featured option, which may be appropriate if you already use Video.js and want a turnkey solution.</td>
<td>Yes</td>
<td>Yes (Video.js Skins)</td>
</tr>
<tr>
<td>Amazon IVS Player Provider for JW Player (p. 62)</td>
<td>A full-featured option, which may be appropriate if you already use JW Player and want a turnkey solution.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Working With Content Security Policy

The Amazon IVS Web player SDK is configured to work on pages that use Content Security Policy (CSP). A few key CSP directives must be in place. Here, we describe a minimal set of directives that are necessary. Additional directives and sources are likely necessary, depending on your specific setup.

### Hosting Assets on the Same Origin

If the SDK library assets are hosted on the same origin as your page, the following directives are necessary:

```
media-src blob;,
connect-src 'self' *.live-video.net;
script-src 'self';
```

Include 'self' in the directive if your CSP policy includes a worker-src directive.

### Hosting Assets on a Separate Origin

If the library assets are hosted on a separate origin from your page, and assuming, for illustration, that origin is `resources.example.com`, additional directives are necessary.

For all browsers except Safari:

```
worker-src blob;,
media-src blob;,
connect-src resources.example.com *.live-video.net;
script-src resources.example.com;
```

For Safari (which does not support the worker-src or the blob: protocol for the script-src directive):

```
media-src blob;,
connect-src resources.example.com *.live-video.net;
script-src 'unsafe-eval' resources.example.com;
```

**Note:** Due to Safari's limited CSP support, it requires a very lenient CSP policy (one including unsafe-eval). This works against the whole point of CSP, to limit dangerous JavaScript from running on a page. With Safari, to avoid having dangerous JavaScript run on your page, we recommend that you host the library assets on the same origin as your page and use our same-origin CSP policy (above).
Known Issues and Workarounds

Please report all issues to Amazon IVS Support.

- When playing recorded content (also known as VOD) on an iOS mobile browser (e.g. Safari or Chrome), seeking backwards will mute the player.
  
  **Workaround:** Call `player.setMuted(false)` after seeking.

- When playing recorded content on an iOS mobile browser, seeking backwards works intermittently when directly selecting the desired position.
  
  **Workaround:** Drag the seek bar to the desired position.

- When playing recorded content on an iOS mobile browser, `player.seekTo()` calls do not work.
  
  **Workaround:** Set `currentTime` on the video HTML element after the `loadeddata` event. For example:

  ```javascript
  videoEl.addEventListener('loadeddata', () => {
    videoEl.currentTime = 30; // seek 30s from the beginning
  });
  ```

- When playing recorded content on an iOS mobile browser using the Video.js integration, the replay button does not work properly.
  
  **Workaround:** Hide the replay button when initializing Video.js. See [https://docs.videojs.com/tutorial-components.html#play-toggle](https://docs.videojs.com/tutorial-components.html#play-toggle).

- When playing a live stream or recorded content on an iOS mobile browser, captions may not be rendered in different sizes and may be re-rendered multiple times.
  
  **Workaround:** None.

- When playing a live stream or recorded content on an iOS mobile browser, `player.getQualities()` calls do not return the list of available qualities.
  
  **Workaround:** None. The player supports only auto-quality mode on iOS browsers.

- When playing a live stream or recorded content on an iOS mobile browser, `player.getLiveLatency()` calls return 0.
  
  **Workaround:** None.

- When playing a live stream on a Google Pixel 4 or 4a mobile browser, playback may stop unexpectedly.
  
  **Workaround:** None.

- When native HTML5 controls are enabled, calls to `setQuality()` are ignored.
  
  **Workaround:** Disable HTML5 controls before calling `player.setQuality()`.

Amazon Interactive Video Service Player: SDK for Android Guide

The Amazon Interactive Video Player (IVS) Android player SDK provides the interfaces required to use the Amazon IVS player on Android.

The `com.amazonaws.ivs.player` package implements the interface described in this document. The following operations are supported:
• Set up (initialize) a player.
• Manage playback.
• Manage quality.
• Receive events.
• Receive errors.

Latest version of Android player: 1.4.1 (Release Notes)

Reference documentation: For information on the most important methods available in the Amazon IVS Android player, see the reference documentation at https://aws.github.io/amazon-ivs-player-docs/1.4.1/android/.

Sample code: See the Android sample repository on GitHub: https://github.com/aws-samples/amazon-ivs-player-android-sample.

Platform requirements: Android 5.0 (Lollipop) or greater is required for development.

Getting Started

Install the Library

To add the Amazon IVS Android player library to your Android development environment, add the library to your module's build.gradle file, as shown here (for the latest version of the Amazon IVS player).

repositories {
    jcenter()
    mavenCentral()
}
dependencies {
    implementation 'com.amazonaws:ivs-player:1.4.1'
}

Create the Player and Set Up Event Listener

The player interface is com.amazonaws.ivs.player.Player. Initialize it as shown below:

```
// Create a player instance
// <this> refers to the current Android PlayerActivity class
player = Player.Factory.create(this);

// Set up to receive playback events and errors
player.addListener(this);
```

Alternately, initialize by using PlayerView:

```
// Create a player instance
// <this> refers to the current Android PlayerActivity class
PlayerView playerView = new PlayerView(this);
Player player = playerView.getPlayer();
// Set up to receive playback events and errors
player.addListener(this);
```

Note: The listener callback methods are executed in the main thread of your Android application.
Set the Surface View for Video

If not using PlayerView, add a SurfaceView to your Android UI layout for displaying a video. This Surface must be available before you can play any video streams. You can access the underlying surface through the SurfaceHolder interface, which is retrieved by calling getHolder(). (See SurfaceView in the Android developer reference). Use the SurfaceHolder.Callback to receive events about surface changes (see SurfaceHolder.Callback).

```java
surfaceView = (SurfaceView) findViewById(R.id.surfaceView);
surfaceView.getHolder().addCallback(this);

@Override
public void surfaceCreated(SurfaceHolder holder) {
    this.surface = holder.getSurface();
    if (player != null) {
        player.setSurface(this.surface);
    }
}

@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    this.surface = null;
    if (player != null) {
        player.setSurface(null);
    }
}
```

Play a Stream

Because the stream is loaded asynchronously, the player must be in a READY state before your application can call the play method to begin playback. Use the Player.Listener interface to determine when the player is in the correct state.

See the following sample code:

```java
player.load(Uri.parse(url));

@Override
public void onStateChanged(Player.State state) {
    switch (state) {
        case BUFFERING:
            // player is buffering
            break;
        case READY:
            player.play();
            break;
        case IDLE:
            break;
        case PLAYING:
            // playback started
            break;
    }
}
```

Release the Player

The player.release() method must be called when the player is no longer in use, to free the resources used by the library. Typically this is done in the onDestroy callback of the Activity or Fragment containing the player.
@Override
protected void onDestroy() {
    super.onDestroy();
    player.removeListener(this);
    player.release();
}

After the `player.release()` method is called the player can no longer be used.

Permissions

The Android player SDK requires the following permission:

<pre><code>&lt;uses-permission android:name="android.permissionINTERNET" /&gt;
</code></pre>

In addition, these optional permissions can improve the playback experience:

<pre><code>&lt;uses-permission android:name="android.permissionACCESS_NETWORK_STATE" /&gt;
&lt;uses-permission android:name="android.permissionACCESS_WIFI_STATE" /&gt;
</code></pre>

Thread Safety

The player API is not thread safe. You should create and use a player instance from one thread; typically this is the application main thread.

SDK Size

The Amazon IVS player SDKs are designed to be as lightweight as possible. For current information about SDK size, see the Release Notes (p. 142).

Important: When evaluating size impact, the size of the AAB/APK produced by Android Studio is not representative of the size of your app downloaded to a user's device. The Google Play Store performs optimizations to reduce the size of your app. We recommend that you use Android App Bundles to serve optimized apps for each device configuration.

Known Issues and Workarounds

Please report all issues to Amazon IVS Support.

- The Android player SDK has a runtime dependency on OkHttp version 4.x. Using OkHttp version 3.x may cause instability or crashes due to an API signature mismatch and OkHttp backwards compatibility issues. Specifically, the player depends on OkHttp version 4.2.2, but it should be compatible with any 4.x version.

  **Workaround:** Use a 4.x version of OkHttp or remove OkHttp from your application.

- When using an Android 11 (API level 30) emulator, you may experience video-layout issues (specifically, zooming of the stream).

  **Workaround:** Play back on the real device instead.
Amazon Interactive Video Service Player: SDK for iOS Guide

The Amazon Interactive Video Service (IVS) iOS player provides the interfaces required to use the Amazon IVS player on iOS.

**Latest version of iOS player:** 1.4.1 ([Release Notes](https://aws.github.io/amazon-ivs-player-docs/1.4.1/ios/))

**Reference documentation:** For information on the most important methods available in the Amazon IVS iOS player, see the reference documentation at [https://aws.github.io/amazon-ivs-player-docs/1.4.1/ios/](https://aws.github.io/amazon-ivs-player-docs/1.4.1/ios/).

**Sample code:** See the iOS sample repository on GitHub: [https://github.com/aws-samples/amazon-ivs-player-ios-sample](https://github.com/aws-samples/amazon-ivs-player-ios-sample).

**Platform requirements:** Xcode 11 or greater is required for development. The SDK supports deployment targets of iOS 10 and above, for both physical devices and the iOS Simulator.

### Getting Started

We recommend that you integrate the player SDK via CocoaPods. (Alternately, you can manually add the framework to your project.)

**Recommended: Integrate the Player SDK (CocoaPods)**

Releases are published via CocoaPods under the name `AmazonIVSPlayer`. Add this dependency to your Podfile:

```plaintext
pod 'AmazonIVSPlayer'
```

Run `pod install` and the SDK will be available in your `.xcworkspace`.

**Alternate Approach: Install the Framework Manually**

1. Download the latest version from [https://player.live-video.net/1.4.1/AmazonIVSPlayer.xcframework.zip](https://player.live-video.net/1.4.1/AmazonIVSPlayer.xcframework.zip).
2. Extract the contents of the archive. `AmazonIVSPlayer.xcframework` contains the SDK for both device and simulator.
3. Embed `AmazonIVSPlayer.xcframework` by dragging it into the **Frameworks, Libraries, and Embedded Content** section of the **General** tab for your application target:

![Frameworks, Libraries, and Embedded Content](image)

### Create Player

The player object is `IVSPlayer`. It can be initialized as shown below:
Swift:

```swift
import AmazonIVSPlayer
let player = IVSPlayer()
```

Objective-C:

```objective-c
#import <AmazonIVSPlayer/AmazonIVSPlayer.h>
IVSPlayer *player = [[IVSPlayer alloc] init];
```

## Set Up Delegate

Delegate callbacks provide information on playback state, events, and errors. All callbacks are invoked on the main queue.

Swift | Objective-C:

```swift
// Self must conform to IVSPlayer.Delegate
player.delegate = self
```

```objective-c
// Self must conform to IVSPlayerDelegate
player.delegate = self;
```

## Display Video

The player displays video in a custom layer, `IVSPlayerLayer`. The SDK also provides `IVSPlayerView`, a `UIView` subclass backed by this layer. Use whichever is more convenient for your application's UI.

In both cases, display the video from a player instance by using the `player` property.

Swift:

```swift
// When using IVSPlayerView:
playerView.player = player

// When using IVSPlayerLayer:
playerLayer.player = player
```

Objective-C:

```objective-c
// When using IVSPlayerView:
playerView.player = player;

// When using IVSPlayerLayer:
playerLayer.player = player;
```

## Load a Stream

The player loads the stream asynchronously. Its state indicates when it is ready to play.

Swift:

```swift
player.load(url)
```

Objective-C:
Play a Stream

When the player is ready, use play to begin playback. Use the delegate interface or key-value observing on the state property to observe the state change. Here is an example of the delegate-based approach:

Swift:

```swift
func player(_ player: IVSPlayer, didChangeState state: IVSPlayer.State) {
    if state == .ready {
        player.play()
    }
}
```

Objective-C:

```objective-c
- (void)player:(IVSPlayer *)player didChangeState:(IVSPlayerState)state {
    if (state == IVSPlayerStateReady) {
        [player play];
    }
}
```

Pause On App Backgrounding

The player does not support playback while the app is in the background, but it does not need to be fully torn down. Pausing is sufficient; see the examples below.

Swift:

```swift
override func viewDidLoad() {
    super.viewDidLoad()

    NotificationCenter.default.addObserver(self,
        selector: #selector(applicationDidEnterBackground(_)),
        name: UIApplication.didEnterBackgroundNotification,
        object: nil)
}

@objc func applicationDidEnterBackground(_ notification: NSNotification) {
    playerView?.player?.pause()
}
```

Objective-C:

```objective-c
- (void)viewDidLoad {
    [super viewDidLoad];

    NSNotificationCenter *defaultCenter = NSNotificationCenter.defaultCenter;
    [defaultCenter addObserver:self
        selector:@selector(applicationDidEnterBackground:) name:UIApplication.didEnterBackgroundNotification object:nil];
}

- (void)applicationDidEnterBackground:(NSNotification *)notification {
    [playerView.player pause];
```
Thread Safety

The player API is not thread safe. You should create and use a player instance from the application main thread.

Putting It All Together

The following simple, view-controller snippet loads and plays a URL in a player view. Note that the `playerView` property is initialized from an XIB/Storyboard, and its class is set to `IVSPlayerView` in Interface Builder using the Custom Class section of the Identity Inspector.

Swift:

```swift
import AmazonIVSPlayer

class MyViewController: UIViewController {
    ...
    // Connected in Interface Builder
    @IBOutlet var playerView: IVSPlayerView!

    override func viewDidLoad() {
        super.viewDidLoad()
        NotificationCenter.default.addObserver(self,
            selector: #selector(applicationDidEnterBackground(_:)),
            name: UIApplication.didEnterBackgroundNotification,
            object: nil)
    }

    @objc func applicationDidEnterBackground(_ notification: NSNotification) {
        playerView?.player?.pause()
    }

    // Assumes this view controller is already loaded.
    // For example, this could be called by a button tap.
    func playVideo(url videoURL: URL) {
        let player = IVSPlayer()
        player.delegate = self
        playerView.player = player
        player.load(videoURL)
    }
}

extension MyViewController: IVSPlayer.Delegate {
    func player(_ player: IVSPlayer, didChangeState state: IVSPlayer.State) {
        if state == .ready {
            player.play()
        }
    }
}
```

Objective-C:

```objective-c
// MyViewController.h
@interface MyViewController : UIViewController

@class IVSPlayerView;
@end

// MyViewController.m
@implementation MyViewController
...
```
SDK Size

The Amazon IVS player SDKs are designed to be as lightweight as possible. For current information about SDK size, see the Release Notes (p. 142).

**Important:** When evaluating size impact, the size of the IPA produced by Xcode is not representative of the size of your app downloaded to a user's device. The App Store performs optimizations to reduce the size of your app.

**Known Issues and Workarounds**

Please report all issues to Amazon IVS Support.

- iOS 10 devices may experience a crash when returning from background.

  **Workaround:** Set the layer's `player` property to `nil` before entering the background.
Amazon Interactive Video Service Player: Video.js Integration

This document describes the most important functions available in the Amazon Interactive Video Service (IVS) Video.js player.

**Latest version of Video.js player integration: 1.4.1** *(Release Notes)*

## Getting Started

Amazon IVS support for Video.js is implemented through a Video.js tech. We provide support through script tags as well as through an npm module. Amazon IVS supports Video.js 7.6.6 and more recent 7.* versions.

Note that when instantiating the player, the Video.js sources option is not supported. Instead, instantiate the player normally and call the Video.js `src()` function. If autoplay is enabled, the stream will start playing; otherwise, use `play()` to start playback.

### Demo

The following live demo shows how to use the Video.js integration with script tags from our Content Delivery Network: [Amazon IVS Player Video.js integration](#).

### Setup With Script Tag

To set up the Amazon IVS tech using the script tag:

1. Include the following tag (for the latest version of the player integration).

   ```html
   <script src="https://player.live-video.net/1.4.1/amazon-ivs-videojs-tech.min.js">
   </script>
   ``

2. Register the tech using the `registerIVSTech` function:

   ```javascript
   registerIVSTech(videojs);
   ```

   where `videojs` is the object provided by Video.js.

3. When creating an instance of the player, add `AmazonIVS` as your first tech in the `techOrder` option.

   When instantiating the player, the Video.js sources option is not supported. Instead, to set the source, instantiate the player normally, then call the Video.js `src()` function on it. If autoplay is enabled, the stream will start playing; otherwise, use `play()` to start playback.

### Sample Code

In this example, `PLAYBACK_URL` is the source stream you want to load. The example uses the latest version of the Amazon IVS Player.

```html
<!doctype html>
<html lang="en">
<head>
</head>
```

---

53
Setup With NPM

To use Amazon IVS player through npm:

1. Install the video.js npm package or ensure that your project has some other access to the Video.js library.

2. Install the amazon-ivs-player npm package:

   
   ```bash
   npm install amazon-ivs-player
   ```

3. When you’re ready to register the Amazon IVS tech, import the registerIVSTech function:

   ```javascript
   import { registerIVSTech } from 'amazon-ivs-player';
   ```

4. Register the tech using the registerIVSTech function:

   ```javascript
   registerIVSTech(videojs, options);
   ```

   where:
   - `videojs` is the object provided by Video.js.
- **options** is the options for the Amazon IVS tech layer. Supported options are:
  - **wasmWorker**: URL where the `amazon-ivs-wasmworker.min.js` file is hosted.
  - **wasmBinary**: URL where the `amazon-ivs-wasmworker.min.wasm` file is hosted.

  The worker files are in your `node_modules/` folder under `amazon-ivs-player/dist/`. You need to host them, to use the IVS player.

5. When creating an instance of the player, add `AmazonIVS` as your first tech in the `techOrder` option:

```javascript
const player = videojs('videojs-player', {
    techOrder: ['AmazonIVS'],
});
```

### TypeScript

If you’re using TypeScript, our npm package includes the following types you may want to import and use.

- **VideoJSEvents**, which describes the returned structure from `getIVSEvents()`.
- **VideoJSIVSTech**, which describes the interface to a player instance that uses the AmazonIVS tech. This can be **intersected** with the `VideoJsPlayer` type exposed by the `@types/video.js` npm package.
- **TechOptions**, which describes the interface defining the configuration options you can send to `registerIVSTech()`.

For more information on these types, see the [Amazon IVS Player: SDK for Web Reference](#).

### Events

To listen to standard Video.js events, use the `on` function of the Video.js player.

To listen to events that are specific to Amazon IVS, add and remove event listeners on the Amazon IVS Web player:

```javascript
player.getIVSPlayer().addEventListener(event, callback);
player.getIVSPlayer().removeEventListener(event, callback);
```

where `callback` is a callback you define, and `event` is one of: `PlayerEventType`, `MetadataEventType`, or `PlayerState`. For more information about events, see the [Amazon IVS Player: SDK for Web Reference](#).

### Errors

For general Video.js errors, listen to the generic `error` event on the player:

```javascript
player.on("error", callback);
```

For errors specific to Amazon IVS, listen on the Amazon IVS player for its own errors:

```javascript
let playerEvent = player.getIVSEvents().PlayerEventType;
player.getIVSPlayer().addEventListener(playerEvent.ERROR, callback);
```

The callback will receive an object with the following fields:
### Field | Description
--- | ---
type | The error type. Corresponds to ErrorType events. For more information, see [Amazon IVS Player: SDK for Web Reference](#).
code | The error code.
source | Source of the error.
message | Human readable error message.

## Plugins

We provide a plugin that creates a UI toggle for available qualities. To use this plugin, it must be loaded by including the `amazon-ivs-quality-plugin.min.js` file if you are using our tech through the following script tag (for the latest version of the IVS Player):

```html
<script src="https://player.live-video.net/1.4.1/amazon-ivs-quality-plugin.min.js"></script>
```

If you are using npm, import the `registerIVSQualityPlugin` from the `amazon-ivs-player` module:

```javascript
import { registerIVSQualityPlugin } from 'amazon-ivs-player';
```

Then, once you create an instance of your Video.js player, the following calls are required to register and enable it:

```javascript
registerIVSQualityPlugin(videojs); // where videojs is the video.js variable
player.enableIVSQualityPlugin(); // where player is the instance of the videojs player
```

This will create a UI menu button which allows you to select a quality for the stream.

## Plugins and TypeScript

If you're using TypeScript, our npm package includes the `VideoJSQualityPlugin` type that you may want to import and use with our plugin. Plugins essentially are mixins, so this type interface is to be used as an intersection type with the `VideoJSIVSTech` typescript interface.

## Content Security Policy

The Amazon IVS Video.js API is configured to work on pages that use Content Security Policy (CSP). See the section on "Working with Content Security Policy" in the [Amazon IVS Player: SDK for Web Guide](#) (p. 43).

## Functions

### Playback

The Amazon IVS Video.js API supports the necessary interfaces for internal use by the Video.js framework. The client application is not likely to need to use these methods directly, since Video.js does the necessary integration and presents a standard interface. However, if needed, one way to access internal Video.js and Amazon IVS player methods is to use the Video.js player object to get the needed object handle to the tech.
To access the API, retrieve the instance of your Video.js player as you would normally:

```javascript
let player = videojs("videoTagId"); //replace videoTagId with your <video> tag’s id
```

Then you can call functions on that instance.

The following are the subset of Video.js functions that the Amazon IVS tech layer overrides. For the full list of Video.js functions, see the video.js API documentation.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description and Amazon IVS-Specific Information</th>
</tr>
</thead>
</table>
| `currentTime (p. 58)` | Gets or sets the time (in seconds from the beginning). 
Amazon IVS: We do not recommend setting current time for live streams.                                                                                                                     |
| `dispose (p. 58)`   | Deletes the player instance                                                                                                           
Amazon IVS: This also deletes the Amazon IVS tech backend.                                                                                                                                 |
| `duration (p. 59)`   | Returns the duration of the video, in seconds.                                                                                          
Amazon IVS: For live streams, this returns `Infinity`.                                                                                                                                   |
| `load (p. 60)`       | Starts loading the `src()` data.                                                                                                       
Amazon IVS: This is a no-op.                                                                                                                                                              |
| `play (p. 60)`       | Plays the stream that was set up via the `src` call.                                                                                     
Amazon IVS: If a live stream was paused, this plays the live stream from the latest frames, instead of continuing from where it was paused.                                                |
| `playbackRate (p. 61)` | Gets or sets the video-playback rate. 1.0 means normal speed; 0.5, half normal speed; 2.0, two times normal speed; and so on. 
Amazon IVS: On a live stream, a get returns 1, and a set is ignored.                                                                                                                     |
| `seekable (p. 61)`   | Returns the `TimeRanges` of the media that can be seeked to.                                                                            
Amazon IVS: For live streams, calling `end(0)` on the return value (`TimeRange`) returns `Infinity`.                                                                                     |

**Amazon IVS Specific**

The Amazon IVS Video.js tech has additional functions for accessing behaviors specific to Amazon IVS features:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getIVSPlayer (p. 59)</code></td>
<td>Returns the underlying Amazon IVS player instance. The full Amazon IVS Player Web API is available through this instance. We recommend using the basic Video.js playback API as much as possible, and using this function only to access Amazon IVS-specific features. The most common functions you are likely to need to</td>
</tr>
</tbody>
</table>
currentTime

Gets or sets the time (in seconds from the beginning).
Amazon IVS: We do not recommend setting current time for live streams.

Signatures

currentTime
currentTime(time)

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>number</td>
<td>If time is absent, gets the current time. If time is present, sets video playback to that time.</td>
</tr>
</tbody>
</table>

Return Value

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The current time, in seconds from the beginning.</td>
</tr>
</tbody>
</table>

dispose

Deletes the player instance.
Amazon IVS: This also deletes the Amazon IVS tech backend.

Signature

dispose()

Parameters

None

Return Value

None
duration

Returns the duration of the video, in seconds.

Amazon IVS: For live streams, this returns Infinity.

Signature

duration()

Parameters

None

Return Value

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The duration of the stream, in seconds. For live streams, this value is Infinity.</td>
</tr>
</tbody>
</table>

getIVSEvents

Returns an object that holds Amazon IVS-specific enums. This is used for listening to Amazon IVS-specific errors and events. For more information, see:

- Events (p. 55) and Errors (p. 55) in this document.
- Amazon IVS Player: SDK for Web Reference for more information about events, error types, and error sources.

Signature

getIVSEvents()

Parameters

None

Return Value

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object with PlayerEventType, PlayerState, and ErrorType keys, which map to their associated enums.</td>
</tr>
</tbody>
</table>

getIVSPlayer

Returns the underlying Amazon IVS player instance. The full Amazon IVS Player Web API is available through this instance. We recommend using the basic Video.js playback API as much as possible,
and using this function only to access Amazon IVS-specific features. The most common functions you are likely to need to access on the Amazon IVS player instance are `setQuality()` and `addEventListener()` / `removeEventListener()`.

**Signature**

```javascript
getIVSPlayer()
```

**Parameters**

None

**Return Value**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MediaPlayer</code></td>
<td>The created instance of the player.</td>
</tr>
</tbody>
</table>

**load**

Starts loading the `src()` data.

Amazon IVS: This is a no-op.

**Signature**

```javascript
load()
```

**Parameters**

None

**Return Value**

None

**play**

Plays the stream that was set up via the `src` call.

Amazon IVS: If a live stream was paused, this plays the live stream from the latest frames, instead of continuing from where it was paused.

**Signature**

```javascript
play()
```

**Parameters**

None
Return Value

None

playbackRate

Gets or sets the video-playback rate. 1.0 means normal speed; 0.5, half normal speed; 2.0, two times normal speed; and so on.

Amazon IVS: On a live stream, a get returns 1, and a set is ignored.

Signatures

playbackRate

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate</td>
<td>number</td>
<td>The playback rate. Valid values: in the range [0.25, 2.0].</td>
</tr>
</tbody>
</table>

Return Value

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The playback rate.</td>
</tr>
</tbody>
</table>

seekable

Returns the TimeRanges of the media that can be seeked to.

Amazon IVS: For live streams, calling end(0) on the return value (TimeRange) returns Infinity.

Signature

seekable()

Parameter

None

Return Value

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeRange</td>
<td>TimeRange of the media that is available for seeking to.</td>
</tr>
</tbody>
</table>
Amazon Interactive Video Service Player: JW Player Integration

This document describes the most important functions available in the Amazon Interactive Video Service (IVS) JW Player integration.

**Latest version of JW Player integration:** 1.4.1 ([Release Notes](#))

## Getting Started

Amazon IVS support for JW Player is implemented through a Provider. Amazon IVS Provider is supported only on JW Player’s web player. The Provider is loaded through a script tag, and any streams requiring Amazon IVS Provider playback must be tagged with `type: 'ivs'` in the playlist. Amazon IVS supports JW Player version 8.18.4 and later.

### Setup

In these instructions, `JW_PLAYER_DIV` is the name of the `<div>` of your JW Player instance and `IVS_STREAM` is your IVS playback URL. To set up the Amazon IVS Provider and enable playback:

1. Include the following script tag (for the latest version of the player integration; in this case, 1.4.1):

   ```html
   <script src="https://player.live-video.net/1.4.1/amazon-ivs-jw-provider.min.js"></script>
   ```

2. Use the `ivs` type to mark your IVS playlist items. Set the `cast` value in your `setup()` to `null` (since Chromecast is not supported).

   ```javascript
   jwplayer(JW_PLAYER_DIV).setup({
   playlist: [{
   file: IVS_STREAM,
   type: 'ivs',
   }
   ]
   });
   ```

3. If you want a reference to the underlying Amazon IVS Player to make Amazon IVS Player API calls or you want references to Amazon IVS-specific enums for callback handling, add a listener to the `'providerPlayer'` event:

   ```javascript
   jwplayer(JW_PLAYER_DIV).on('providerPlayer', function (player) {
   // player object has 'ivsPlayer' and 'ivsEvents' properties
   // ...callback code...
   });
   ```

## Sample Code

In this example, `JW_PLAYER_LIB` is the URL to your JW Player library script and `IVS_STREAM` is your IVS playback URL.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <script src=JW_PLAYER_LIB></script>
  <script src="https://player.live-video.net/1.4.1/amazon-ivs-jw-provider.min.js"></script>
</head>
```
Events

To listen to standard JW Player events, use the `on` function of the JW Player.

To listen to events that are specific to Amazon IVS, or to add and remove event listeners on the Amazon IVS Web player, you must listen to the `providerPlayer` event to get a reference to the Amazon IVS Player and then add event listening onto it. For example:

```javascript
// store a default value for ivsPlayer
var ivsPlayer = {};

// store references to the Amazon IVS Player and Amazon IVS Events:
jwplayer(JW_PLAYER_DIV).on('providerPlayer', function (player) {
  ivsPlayer = player.ivsPlayer;
});

// set up event listening
ivsPlayer.addEventListener(event, callback);
ivsPlayer.removeEventListener(event, callback);
```

where callback is a callback that you define, and event is one of: `PlayerEventType`, `PlayerState`, or `ErrorType`. For more information about events, see the Amazon IVS Player: SDK for Web Reference.

The `providerPlayer` event is emitted by JW Player, and the callback you register with it will receive an object with the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ivsPlayer</td>
<td>Returns the underlying Amazon IVS player instance. The full Amazon IVS Player Web API is available through this instance. We recommend using the basic JW</td>
</tr>
</tbody>
</table>
Errors

For general JW Player errors, use the on function of the JW Player to listen to error events.

For errors specific to Amazon IVS, listen on the Amazon IVS player for its own errors:

```javascript
// set default values for ivsPlayer and ivsEvents
var ivsPlayer = {};
var ivsEvents = {};

// store references to the Amazon IVS Player and Amazon IVS Events
jwplayer(JW_PLAYER_DIV).on('providerPlayer', function (player) {
    ivsPlayer = player.ivsPlayer;
    ivsEvents = player.ivsEvents;
});

// set up event listening:
let playerEvent = ivsEvents.PlayerEventType;
ivsPlayer.addEventListener(playerEvent.ERROR, callback);
```

The callback will receive an object with the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>The error type. Corresponds to ErrorType events. For more information, see Amazon IVS Player: SDK for Web Reference.</td>
</tr>
<tr>
<td>code</td>
<td>The error code.</td>
</tr>
<tr>
<td>source</td>
<td>Source of the error.</td>
</tr>
<tr>
<td>message</td>
<td>Human readable error message.</td>
</tr>
</tbody>
</table>

Content Security Policy

The Amazon IVS Provider API is configured to work on pages that use Content Security Policy (CSP). See the section on "Working with Content Security Policy" in the Amazon IVS Player: SDK for Web Guide (p. 43).

Limitations

The Provider does not support casting. If you enabled casting in the JW Player dashboard, you can disable it by setting cast to null when calling setup(). This hides the casting button.
Embedding Metadata within a Video Stream

Amazon Interactive Video Service (IVS) timed metadata provides a way to embed metadata in an Amazon IVS stream. It ensures that all your viewers receive the metadata at the same time in the video stream, regardless of stream latency or geographic location.

What is Timed Metadata?

Timed metadata is metadata with timestamps. It can be inserted into a stream programmatically, using the Amazon IVS API. When Amazon IVS processes a stream, the timed metadata is synchronized with the audio and video frames. During playback, all viewers of the stream get the metadata at the same time relative to the stream. The timecode serves as a cue point, which can be used to trigger an action based on the data, such as the following:

- Updating player statistics for a sports stream.
- Sending product details for a live shopping stream.
- Sending questions for a live quiz stream.

Amazon IVS timed metadata uses ID3 tags embedded in the video segments. As a result, they are available in the recorded video.

Setting Up IAM Permissions

Prerequisite: Before proceeding, you should have stepped through Getting Started with Amazon IVS (p. 3) (including creating an IAM user and setting up permissions).

Next, you must give your IAM user permission to use timed metadata. Follow these steps:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Users, then choose the desired user (the user name you specified when you created an AWS account).
3. In the user Summary window, on the Permissions tab, choose Add inline policy (on the right side).
4. On the JSON tab, paste in this blob:

   ```json
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "ivs:PutMetadata"
               ]
           }
       ]
   }
   ```
5. Still in the Create Policy window, choose Review Policy. Give the policy a Name, then choose Create Policy.

6. You’re returned to the user Summary window, showing your new policy name.

Inserting Timed Metadata

You can insert timed metadata only into an active stream on a specified channel.

Using the AWS CLI

For testing, the easiest way to add timed metadata is with the AWS CLI. Using the AWS CLI requires that you first download and configure the CLI on your machine. You may have already done that when you stepped through Getting Started with Amazon IVS (p. 3); if not, do it now. For details, see the AWS Command Line Interface User Guide.

Once you have the CLI:

1. Run the put-metadata command and pass in the channel ARN and your metadata:

   aws ivs put-metadata --channel-arn <your-channel-arn> --metadata <your-metadata>

   For example:

   aws ivs put-metadata --channel-arn arn:aws:ivs:us-west-2:465369119046:channel/GbiYJna5hFoC --metadata '{"question": "What does IVS stand for?", "correctIndex": 0, "answers": ["interactive video service", "interesting video service", "ingenious video service"]}'

2. Amazon IVS checks whether the stream is live. If the stream is not live, you get an error; otherwise, the CLI returns without an error and the metadata (text blob) is inserted into the stream. This happens as soon as possible. There is no guarantee as to when this occurs; however, all viewers see the metadata at the same point in the stream.

Using the Amazon IVS API

To programmatically insert timed metadata, use the PutMetadata endpoint, described in the Amazon IVS API Reference.

Here is an example HTTP request:

POST /PutMetadata HTTP/1.1
{
   "channelArn": "my_channel",
   "metadata": "{"question": "What does IVS stand for?", "correctIndex": 0, "answers": ["interactive video service", "interesting video service", "ingenious video service"]}"}
Consuming Timed Metadata

Use the Amazon IVS Player to consume timed metadata embedded in a video stream. See Amazon IVS Player (p. 38) and the rest of the Player documentation.

Below are example snippets that print any metadata received to the console using the Amazon IVS Player SDK. An event is triggered whenever playback reaches a segment with embedded metadata. (The event is TEXT_METADATA_CUE for Web, onCue() for Android, and player(_:didOutputCue:) for iOS.) You can use this event to initiate functionality within your client application, such as updating an interactive widget. This event is triggered for both live and recorded content.

Amazon IVS Player SDK for Web:

```javascript
const player = IVSPlayer.create();
player.addEventListener(IVSPlayer.PlayerEventType.TEXT_METADATA_CUE,
    function (cue) {
        console.log('Timed metadata: ', cue.text);
    });
```

Amazon IVS Player SDK for Android:

```java
@Override
public void onCue(@NonNull Cue cue) {
    if(cue instanceof TextMetadataCue) {
        Log.i("Timed Metadata: ", ((TextMetadataCue)cue).text);
    }
}
```

Amazon IVS Player SDK for iOS:

```swift
func player(_ player: IVSPlayer, didOutputCue cue: IVSCue) {
    if let textMetadataCue = cue as? IVSTextMetadataCue {
        print("Timed Metadata: \(textMetadataCue.text)"
    }
}
```

Note: Timed metadata is supported for iOS Safari and iOS Chrome in Player 1.3.0 and later.

Sample Demo: Quiz App

Code samples of an interactive quiz app are available on GitHub. We use JSON via timed metadata to populate a quiz UI to display questions and answers. The answers are selectable and reveal whether the selection is correct.

<table>
<thead>
<tr>
<th>Amazon IVS Player SDK</th>
<th>Repo of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td></td>
</tr>
<tr>
<td>Web</td>
<td><a href="https://github.com/aws-samples/amazon-ivs-basic-web-sample">https://github.com/aws-samples/amazon-ivs-basic-web-sample</a></td>
</tr>
<tr>
<td></td>
<td>Within this repo, see the quiz demo (and live demo).</td>
</tr>
<tr>
<td>Android</td>
<td><a href="https://github.com/aws-samples/amazon-ivs-player-android-sample">https://github.com/aws-samples/amazon-ivs-player-android-sample</a></td>
</tr>
<tr>
<td></td>
<td>Within this repo, see the quiz demo.</td>
</tr>
</tbody>
</table>
Viewing Timed Metadata

If desired, you can view the timed metadata embedded in your live stream, in the console:

1. Open the Amazon IVS console.
2. In the top left, choose the hamburger icon to open the navigation pane, then choose Live channels.
3. Choose the channel whose stream you want to view, to go to a details page for that channel.
   
   The live stream is playing in the Live stream section of the page.
4. At the bottom of the window, choose Timed Metadata.
   
   While the player is playing, as each timed-metadata event is received, its value and time received are displayed.

For More Information

See Using Amazon Interactive Video Service Timed Metadata, the first of a two-part blog series on using Amazon IVS timed metadata.
Setting Up Private Channels

Amazon Interactive Video Service (IVS) offers customers the ability to create private channels, allowing customers to restrict their streams by channel or viewer. Customers control access to video playback by enabling playback authorization on channels and generating signed JSON Web Tokens (JWTS) for authorized playback requests.

Requiring playback authorization on a channel is optional. When a viewer tries to watch a stream, if the channel has authorization enabled, Amazon IVS verifies that the viewer has a valid playback token in the request. A playback token is a JWT that the Amazon IVS customer signs (with a playback authorization key) and includes with every playback request for a channel that has playback authorization enabled.

Topics
- Workflow for Private Channels (p. 69)
- Create or Import a Playback Key (p. 70)
- Enable Playback Authorization on Channels (p. 71)
- Generate and Sign Playback Tokens (p. 73)
- List Playback Keys (p. 74)
- Delete Playback Keys (p. 75)
- Get Information about Playback Keys (p. 76)

Workflow for Private Channels

1. When a viewer tries to load the webpage for a private stream, the browser requests an access token. (The customer provides the browser code to do this.)
2. The customer’s backend app receives the access-token request and determines whether that viewer should be authorized to view the stream. If yes, the backend generates a JWT, uses the customer’s private key to sign it, and returns the signed JWT in a playback request to the browser.
3. The browser loads the stream, using a request to the Amazon IVS player (or other player) SDK. The request contains the stream playback URL and the signed JWT.
4. Amazon IVS uses the customer's public key to verify that the JWT was signed using the correct private key.
5. If the JWT is verified, Amazon IVS plays the private stream for the viewer.

Customers are responsible for creating:

- The browser code to request access tokens.
- The backend server app that generates and signs JWTs.
- A playback authorization key pair. This has two parts: a public key that AWS retains and a private key that you download. With the private key, you sign the JWTs that authorize access to your private channel.

The method described above — using a network request from the browser to fetch tokens — is not the only way to implement playback authorization. Alternately, customers could send the signed playback tokens in the initial webpage, to reduce the number of network round trips that a viewer needs to make.

In the sections below, we describe how to make a channel private (enable playback authorization), generate and sign playback tokens, and work with playback key pairs.

Note: In the console instructions below, if the left navigation menu is not displaying, you can open it by choosing the hamburger icon in the top left.

### Create or Import a Playback Key

Amazon IVS allows a maximum of three key pairs that can be used to sign and verify playback tokens. Amazon IVS does not offer any key rotations.

**Once imported, playback keys cannot be updated.** Instead, you must delete the existing playback key and import a new key.

You need to generate an [ECDSA public/private key pair](https://docs.aws.amazon.com/ivs/latest/userguide/articles/keys.html) to sign the JWTs and upload the public key to Amazon IVS as a playback-key resource. Then Amazon IVS can verify the signature in playback requests.

### Console Instructions

To create a key pair:

1. Open the [Amazon IVS console](https://console.aws.amazon.com/ivs). Choose your channel's region if you are not already on it.
2. In the left navigation menu, choose **Playback keys**.
3. Choose **Create playback key**. A **Create playback key** dialog appears.
4. Follow the prompts and choose **Create**.
5. Amazon IVS generates a new key pair. The public key is imported as a playback key resource and the private key is immediately made available for download.

   Amazon IVS generates the key on the client side and does not store the private key. **Be sure you save the key; you cannot retrieve it later.**

To import an existing public key:

1. Open the [Amazon IVS console](https://console.aws.amazon.com/ivs). Choose your channel's region if you are not already on it.
2. In the left navigation menu, choose **Playback keys**.
3. Choose **Import**. An **Import playback key** dialog appears.
4. Follow the prompts and choose **Import**.
5. Amazon IVS imports your public key and generates a playback key resource.

### CLI Instructions

There are various methods to create a key pair. For example, here is how to generate a P384 EC key with OpenSSL (you may have to install **OpenSSL** first):

```
openssl ecparam -name secp384r1 -genkey -noout -out priv.pem
openssl ec -in priv.pem -pubout -out public.pem
```

After generating the public key, import it into Amazon IVS:

```
aws ivs import-playback-key-pair --public-key-material "`cat public.pem`" --region <aws-region>
```

You can omit `--region <aws-region>` if the region is in your local AWS configuration file.

Here is an example response:

```
{
  "keyPair": {
    "tags": {}
  }
}
```

### API Request

```
POST /ImportPlaybackKeyPair HTTP/1.1
{
  "publicKeyMaterial": "<pem file contents>"
}
```

### Enable Playback Authorization on Channels

A channel's authorization requirement can be configured when the channel is created or later (using an update endpoint). Note that the steps are the same whether you want to enable or disable playback authorization.

#### Console Instructions

To enable authorization when creating a channel:

1. Open the **Amazon IVS console**. Choose your channel's region if you are not already on it.
2. In the **Get started** box (top right), choose **Create channel**.
3. On the Channel create page, choose Custom configuration.
4. In the Playback authentication section, turn on Enable token-authentication requirement for video playback.
5. Follow the rest of the prompts to create a channel. (See Getting Started with Amazon IVS (p. 3).)

To enable authorization by updating an existing channel:
1. Open the Amazon IVS console. Choose your channel's region if you are not already on it.
2. In the left navigation menu, choose Channels.
3. Choose the checkbox for the channel you want to update, then choose Edit.
4. In the Playback authentication section, turn on Enable token-authentication requirement for video playback.
5. Click Save changes.

CLI Instructions

To enable authorization when creating a channel:

```bash
aws ivs create-channel --authorized --region <aws-region>
```

You can omit --region <aws-region> if the region is in your local AWS configuration file.

Here is an example response. Note that authorized is true.

```json
{
  "streamKey": {
    "channelArn": "arn:aws:vs:us-west-2:123456789:channel/fbc789c1-2c56-4ce6-a30a-d99275dc4481",
    "value": "sk_us-west-2_abcd1234efgh5678ijkl",
    "arn": "arn:aws:vs:us-west-2:123456789:stream-key/62f15f1b-fe31-4127-b252-066ac7f55a7",
    "tags": {}
  },
  "channel": {
    "name": "test-channel",
    "tags": {},
    "authorized": true,
    "latencyMode": "LOW",
    "ingestEndpoint": "jds34ksgd3las.global-contribute.live-video.net",
    "playbackUrl": "https://b37c565f6d79.us-west-2.playback.live-video.net/api/video/v1/aws.ivs.us-west-2.123456789.channel.0U4KS4LA1Dx.m3u8",
    "arn": "arn:aws:ivs:us-west-2:123456789:channel/fbc789c1-2c56-4ce6-a30a-d99275dc4481"
  }
}
```

To enable authorization by updating an existing channel:

```bash
aws ivs update-channel --arn
--authorized
```

This is just an example; you must specify your own channel ARN after --arn. As when creating a channel, authorized is true in the update response.
API Request (Create and Update)

```
POST /CreateChannel HTTP/1.1
{
  "name": "<your channel name>",
  "authorized": true
}
```

```
POST /UpdateChannel HTTP/1.1
{
  "arn": "<channel arn>",
  "authorized": true
}
```

Generate and Sign Playback Tokens

For details on working with J Torres and the supported libraries for signing tokens, visit jwt.io.

**Token Schema**

All JWTs have three fields: header, payload, and signature.

- **The header** specifies:
  - `alg` is the signing algorithm. This is ES384, an ECDSA signature algorithm that uses the SHA-384 hash algorithm.
  - `typ` is the token type, JWT.

  ```
  {
    "alg": "ES384",
    "typ": "JWT"
  }
  ```

- **The payload** contains data specific to Amazon IVS:
  - `channel-arn` is a reference for the video-playback request.
  - `access-control-allow-origin` is an optional field that can be used to restrict playback to specified domains; i.e., to make a stream viewable from only a specified website. For example, you may want to prevent people from embedding the player on other websites. By default, playback is allowed on all domains.
  - `exp` is a Unix timestamp for when the token expires. This does not indicate the length of time that the stream can be viewed. The token is validated when the viewer initializes playback, not throughout the stream. Enter this value as an integer type value.

  ```
  {
    "aws:channel-arn": "<channel_arn>",
    "aws:access-control-allow-origin": "<your-website>",
    "exp": <unix timestamp>
  }
  ```

- **To create the signature**, use the private key with the algorithm specified in the header (ES384) to sign the encoded header and encoded payload.

  ```
  ECDSASHA384(
    base64UrlEncode(header) + "." +
  ```
Instructions

1. Generate the token's signature with the ES384 signing algorithm and a private key that is associated with one of your playback-key resources (see the ECDSASHA384 example above).
2. Assemble the token.

   ```
   base64UrlEncode(header) + "." + 
   base64UrlEncode(payload) + "." + 
   base64UrlEncode(signature)
   ```

3. Append the signed token to the playback URL as a query parameter.

   ```
   https://b37c565f6d790a14a0e78afaa6808a80.us-west-2.playback.live-video.net/
   api/video/v1/aws.ivs.us-west-2.123456789.
   channel.fbc789c1-2c56-4ce6-a30a-d99275dc4481.m3u8?token=<token>
   ```

List Playback Keys

Amazon IVS customers can get a list of all of their playback-key resources at any time.

Console Instructions

1. Open the Amazon IVS console. Choose your channel's region if you are not already on it.
2. In the left navigation menu, choose Playback keys.

   All playback-key resources associated with your account are displayed. Deleted keys are not displayed, and there is no history of past keys.

Note: You also can get to the Playback keys page from the Amazon IVS console home page, by choosing Playback keys in the left navigation menu.

CLI Instructions

```
aws ivs list-playback-key-pairs --region <aws-region>
```

You can omit --region <aws-region> if the region is in your local AWS configuration file.

Example response:

```json
{
   "keyPairs": [
   {
      "arn": "arn:aws:ivs:us-west-2:991729659840:playback-key/3db9fc15-df57-4c02-
      b5a6-d4ee3448b8ad",
      "tags": {}
   },
   ]
}
```
API Request

For usage information, see ListPlaybackKeyPairs in the Amazon IVS API Reference.

POST /ListPlaybackKeyPairs HTTP/1.1
{
   "maxResults": number,
   "nextToken": "string"
}

Delete Playback Keys

Amazon IVS customers can delete playback keys from their accounts. Deleted keys will remove the resource from the customer's account; playback tokens signed with deleted keys will not pass verification.

Console Instructions

1. Open the Amazon IVS console. Choose your channel's region if you are not already on it.
2. In the left navigation menu, choose Playback keys.
3. Choose the key(s) you want to delete.
4. Choose Delete. A Delete playback key dialog appears.
5. Choose Delete playback key.

Note: You also can get to the Playback keys page from the Amazon IVS console home page, by choosing Playback keys in the left navigation menu.

CLI Instructions

You can delete playback keys via the AWS CLI, if you have the key's ARN. Amazon IVS does not support batch deletes via the CLI.

```
aws ivs delete-playback-key-pair --arn arn:aws:ivs:us-west-2:991729659840:playback-key/3ff88c71-b18e-415f-948b-1bb4e605a97
```

You can omit `--region <aws-region>` if the region is in your local AWS configuration file.

On success, there is no response. You can run the `get` command (below) to verify that the key was deleted.

Here is an example error response:

```
An error occurred (ResourceNotFoundException) when calling the
```
DeletePlaybackKeyPair operation: ResourceNotFoundException:

API Request

POST /DeletePlaybackKeyPair HTTP/1.1
{
    "arn": "<playback key arn>"
}

Get Information about Playback Keys

Amazon IVS customers can get information about their playback key resources. It is important to note that the associated private key will not be available, even in the case that the playback key was created by Amazon IVS via the console.

Console Instructions

1. Open the Amazon IVS console. Choose your channel’s region if you are not already on it.
2. In the left navigation menu, choose Playback keys.
3. Choose the key you want to get more details about and choose View details.

Note: You also can get to the Playback keys page from the Amazon IVS console home page, by choosing Playback keys in the left navigation menu.

CLI Instructions

aws ivs get-playback-key-pair --arn arn:aws:ivs:us-west-2:991729659840:playback-key/3db9fc15-df57-4c02-b5a6-d4ee3448b8ad --region <aws-region>

You can omit --region <aws-region> if the region is in your local AWS configuration file.

Example response:

{
    "keyPair": {
        "fingerprint": "a2:b5:b3:0b:be:8e:73:00:0e:ad:e9:bb:02:c9:81:9a",
        "tags": {}
    }
}

API Request

POST /GetPlaybackKeyPair HTTP/1.1
{
    "arn": "<playback key arn>"
}
Auto-Record to Amazon S3

This section provides information about the auto-record-to-S3 feature of Amazon IVS. We discuss data storage for recorded Amazon IVS streams. We explain the storage contents and metadata file schema. We also discuss playback of your recorded content.

<table>
<thead>
<tr>
<th>For details on ...</th>
<th>See ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting up and stopping video recording</td>
<td>Create a Channel with Optional Recording in Getting Started with Amazon IVS</td>
</tr>
<tr>
<td>The API</td>
<td>IVS API Reference</td>
</tr>
<tr>
<td>Costs</td>
<td>Amazon IVS Costs</td>
</tr>
</tbody>
</table>

S3 Prefix

The S3 prefix is a unique directory structure for each live stream that is recorded. All media and metadata files for the live stream are written within this directory. For channels with recording enabled, the S3 prefix is generated when a live session starts and will be provided in the CloudWatch event at the start and end of a recording.

The S3 prefix has the following format:

/ivs/v1/<aws_account_id>/<channel_id>/<year>/<month>/<day>/<hours>/<minutes>/<recording_id>

Where:

- **aws_account_id** is the ID of your AWS account (generated when you created an AWS account), from which the channel is created.
- **channel_id** is the resource ID part of the channel ARN (the last part of the Amazon Resource Name). See ARN in the Glossary (p. 124).
- **<year>/<month>/<day>/<hours>/<minutes>** is a UTC timestamp when recording starts.
- **recording_id** is a unique ID generated for each recording session.

For example:

/ivs/v1/123456789012/AsXego4U6tnj/2020/6/23/20/12/j8Z9O91ndcVn

Recording Contents

When recording starts, video segments and metadata files are written to the S3 bucket that is configured for the channel. These contents are available for post-processing or playback as on-demand video.

Note that after a live stream starts and the Recording Start EventBridge event is emitted, it takes a little time before the manifest files and video segments are written. We recommend that you play back or process recorded streams only after the Recording End event is sent. (See Using Amazon EventBridge with Amazon IVS (p. 85).)

The following is a sample directory structure and contents of a recording of a live Amazon IVS session:
The `events` folder contains the metadata files corresponding to the recording event. JSON metadata files are generated when recording starts, ends successfully, or ends with failures:

- `events/recording-started.json`
- `events/recording-ended.json`
- `events/recording-failed.json`

A given `events` folder will contain `recording-started.json` and either `recording-ended.json` or `recording-failed.json`.

These contain metadata related to the recorded session and its output formats. JSON details are given below.

The `media` folder contains all supported media contents, in two subfolders:

- `hls` contains all media and manifest files generated during the live session and is playable with the Amazon IVS player.
- `thumbnails` contains thumbnail images generated during the live session. Thumbnails are generated and written to the bucket every minute.

Important: The contents within the `media` folder are dynamically generated and determined by the characteristics of the first received video segments; the folder contents may not represent the ultimate characteristics (e.g., rendition quality). Do not make any assumptions about the static path. To discover the HLS renditions available and its path, use the JSON metadata files described in the following section.

## JSON Metadata Files

When a recording state-change event occurs, a corresponding Amazon CloudWatch metric is generated and a metadata file is written within the S3 prefix. (See Monitoring Amazon IVS with Amazon CloudWatch (p. 93).)

This metadata is in JSON format. It comprises the following information.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel_arn</td>
<td>string</td>
<td>Yes</td>
<td>ARN of the channel broadcasting the live stream.</td>
</tr>
<tr>
<td>media</td>
<td>object</td>
<td>Yes</td>
<td>Object that contains the enumerated objects of media content available for this recording. Valid values: &quot;hls&quot;, &quot;thumbnails&quot;.</td>
</tr>
<tr>
<td>hls</td>
<td>object</td>
<td>Yes</td>
<td>Enumerated field that describes the Apple HLS format output.</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>duration_ms</td>
<td>integer</td>
<td>Conditional</td>
<td>Duration of the recorded HLS content in milliseconds. This is available only when recording_status is &quot;RECORDING_ENDED&quot; or &quot;RECORDING_ENDED_WITH_FAILURE&quot;. If a failure occurred before any recording was done, this is 0.</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>Yes</td>
<td>Relative path from the S3 prefix where HLS content is stored.</td>
</tr>
<tr>
<td>playlist</td>
<td>string</td>
<td>Yes</td>
<td>Name of the HLS master playlist file.</td>
</tr>
<tr>
<td>renditions</td>
<td>object</td>
<td>Yes</td>
<td>Array of renditions (HLS variant) of metadata objects. There always is at least one rendition.</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>Yes</td>
<td>Relative path from the S3 prefix where HLS content is stored for this rendition.</td>
</tr>
<tr>
<td>playlist</td>
<td>string</td>
<td>Yes</td>
<td>Name of the media playlist file for this rendition.</td>
</tr>
<tr>
<td>resolution_height</td>
<td>int</td>
<td>Conditional</td>
<td>Pixel resolution height of the encoded video. This is available only when the rendition contains a video track.</td>
</tr>
<tr>
<td>resolution_width</td>
<td>int</td>
<td>Conditional</td>
<td>Pixel resolution width of the encoded video. This is available only when the rendition contains a video track.</td>
</tr>
<tr>
<td>thumbnails</td>
<td>object</td>
<td>Yes</td>
<td>Enumerated field that describes thumbnails output.</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>Yes</td>
<td>Relative path from the S3 prefix where thumbnail content is stored.</td>
</tr>
<tr>
<td>recording_ended_at</td>
<td>string</td>
<td>Conditional</td>
<td>RFC 3339 UTC timestamp when the recording ended. This is available only when recording_status is &quot;RECORDING_ENDED&quot; or &quot;RECORDING_ENDED_WITH_FAILURE&quot;.</td>
</tr>
<tr>
<td>recording_started_at</td>
<td>string</td>
<td>Yes</td>
<td>RFC 3339 UTC timestamp when the recording started.</td>
</tr>
</tbody>
</table>
### Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>recording_status</td>
<td>string</td>
<td>Yes</td>
<td>Status of the recording. Valid values: &quot;RECORDING_STARTED&quot;, &quot;RECORDING_ENDED&quot;, &quot;RECORDING_ENDED_WITH_FAILURE&quot;.</td>
</tr>
<tr>
<td>recording_status_message</td>
<td>string</td>
<td>Conditional</td>
<td>Descriptive information on the status. This is available only when recording_status is &quot;RECORDING_ENDED&quot; or &quot;RECORDING_ENDED_WITH_FAILURE&quot;.</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
<td>Yes</td>
<td>The version of the metadata schema.</td>
</tr>
</tbody>
</table>

### Example: recording_started.json

```json
{
    "version" : "v1",
    "channel_arn" : "arn:aws:ivs:us-west-2:123456789012:channel/AsXego4U6tnj",
    "recording_started_at" : "2020-06-12T12:53:26Z",
    "recording_status" : "RECORDING_STARTED",
    "media" : {
        "hls" : {
            "path" : "media/hls",
            "playlist" : "master.m3u8",
            "renditions" : [
                {
                    "path" : "480p30",
                    "playlist" : "playlist.m3u8",
                    "resolution_height" : 480,
                    "resolution_width" : 852
                },
                {
                    "path" : "360p30",
                    "playlist" : "playlist.m3u8",
                    "resolution_height" : 360,
                    "resolution_width" : 640
                },
                {
                    "path" : "160p30",
                    "playlist" : "playlist.m3u8",
                    "resolution_height" : 160,
                    "resolution_width" : 284
                },
                {
                    "path" : "720p60",
                    "playlist" : "playlist.m3u8",
                    "resolution_height" : 720,
                    "resolution_width" : 1280
                }
            ]
        },
        "thumbnails" : {
            "path" : "media/thumbnails"
        }
    }
}
```
Example: recording Ended.json

```
{
  "version": "v1",
  "channel_arn": "arn:aws:ivs:us-west-2:123456789012:channel/AsXego4U6tnj",
  "recording_ended_at": "2020-06-14T12:53:20Z",
  "recording_started_at": "2020-06-12T12:53:26Z",
  "recording_status": "RECORDING_ENDED",
  "media": {
    "hls": {
      "duration_ms": 172794489,
      "path": "media/hls",
      "playlist": "master.m3u8",
      "renditions": [
        {
          "path": "480p30",
          "playlist": "playlist.m3u8",
          "resolution_height": 480,
          "resolution_width": 852
        },
        {
          "path": "360p30",
          "playlist": "playlist.m3u8",
          "resolution_height": 360,
          "resolution_width": 640
        },
        {
          "path": "160p30",
          "playlist": "playlist.m3u8",
          "resolution_height": 160,
          "resolution_width": 284
        },
        {
          "path": "720p60",
          "playlist": "playlist.m3u8",
          "resolution_height": 720,
          "resolution_width": 1280
        }
      ]
    },
    "thumbnails": {
      "path": "media/thumbnails"
    }
  }
}
```

Example: recording Failed.json

```
{
  "version": "v1",
  "channel_arn": "arn:aws:ivs:us-west-2:123456789012:channel/AsXego4U6tnj",
  "recording_ended_at": "2020-06-14T12:53:20Z",
  "recording_started_at": "2020-06-12T12:53:26Z",
  "recording_status": "RECORDING_ENDED_WITH_FAILURE",
  "recording_status_message": "InternalServerException",
  "media": {
    "hls": {
      "duration_ms": 172794489,
      "path": "media/hls",
      "playlist": "master.m3u8",
      "renditions": [
        {
          "path": "480p30",
          "playlist": "playlist.m3u8",
          "resolution_height": 480,
          "resolution_width": 852
        },
        {
          "path": "360p30",
          "playlist": "playlist.m3u8",
          "resolution_height": 360,
          "resolution_width": 640
        },
        {
          "path": "160p30",
          "playlist": "playlist.m3u8",
          "resolution_height": 160,
          "resolution_width": 284
        },
        {
          "path": "720p60",
          "playlist": "playlist.m3u8",
          "resolution_height": 720,
          "resolution_width": 1280
        }
      ]
    },
    "thumbnails": {
      "path": "media/thumbnails"
    }
  }
}
```
Discovering the Renditions of a Recording

When you stream content to an Amazon IVS channel, auto-record-to-s3 uses the source video to generate multiple renditions. Using Adaptive Bitrate Streaming (p. 38) (ABR), the Amazon IVS Player automatically switches the renditions (bitrates) as needed to optimize playback for varying network conditions.

Each rendition generated during live streaming is recorded in a unique path within the S3 recording prefix. The resolution detail, path, and playlist file names are stored in a JSON metadata file during the start and stop of the recording.

Important: Do not make any assumptions about the static rendition path or the list of generated renditions, as these are subject to change. Do not assume that a specific rendition will always be available for an Amazon IVS recording. To determine the available renditions, resolutions, and paths, refer to the metadata files.

The event/recording_started.json or event/recording_ended.json file within the recording prefix contains the paths and names of media files within the recording prefix. All path elements are relative to the previous path in the hierarchy. Elements under media > hls describe HLS assets, with master playlist name and path defined at this level.

Here is a Python code snippet that shows how to generate a master playlist path using the S3 recording prefix and metadata file:

```
def get_master_playlist(metadata_json, s3_recording_prefix):
    return s3_recording_prefix + '/' + metadata_json['media']['hls']['path'] + '/' + metadata_json['media']['hls']['playlist']
```

Elements under media > hls > renditions describe the list of renditions recorded. The resolution_height and resolution_width properties can be used to identify the video resolution. The path and playlist elements can be used to derive the rendition playlist path. Use these fields to determine which rendition to use for any post processing.

To discover the highest available rendition playlist for a recording, you can subscribe to "IVS Recording State Change" EventBridge events. (See Using Amazon EventBridge with Amazon IVS (p. 85). Below is a sample Python script that illustrates using a lambda function subscribed to those events.

```
import json
import boto3
s3 = boto3.resource('s3')
```
Playback of Recorded Content from Private Buckets

Objects recorded with the Auto-Record to Amazon S3 feature are private by default; hence, these objects are inaccessible for playback using the direct S3 URL. If you try to open the HLS master manifest (m3u8 file) for playback using the Amazon IVS player or another player, you will get an error (e.g., "You do not have permission to access the requested resource"). Instead, you can play back these files with the Amazon CloudFront CDN (Content Delivery Network).

Amazon CloudFront Distribution

CloudFront distributions can be configured to serve content from private buckets. Typically this is preferable to having openly accessible buckets where reads bypass the controls offered by CloudFront. Your distribution can be set up to service from a private bucket by creating an origin access identity (OAI), which is a special CloudFront user that has read permissions on the private origin bucket. You can create the OAI when you create your distribution or you can add one afterwards, through the CloudFront console or API. See Creating a CloudFront OAI and Adding it to Your Distribution.

Playback from Amazon CloudFront

Once you have set up your distribution using an OAI to gain access to your private bucket, your video files should be available for consumption through the CloudFront URL. Your CloudFront URL is visible on the General tab for that distribution on the AWS console for CloudFront; it will look something like this:
To stream your recorded video through your distribution, find the object key for your `master.m3u8` file. It should be something like this:

```
ivs/v1/012345678912/a0bCDeFGH1IjK/2021/4/20/12/03/aBcdEFghIjkL/media/hls/master.m3u8
```

Append the object key to the end of your CloudFront URL. Your final URL will be something like this:

```
https://a1b23cdef4ghij.cloudfront.net/ivs/v1/012345678912/a0bCDeFGH1IjK/2021/4/20/12/03/
 aBcdEFghIjkL/media/hls/master.m3u8
```

Now you can play back your recorded video as if you were playing directly from a bucket.

For more information, see CloudFront documentation on OAI.s.
Using Amazon EventBridge with Amazon Interactive Video Service

You can use Amazon EventBridge to monitor your Amazon Interactive Video Service (IVS) streams.

Amazon IVS sends change events about the status of your streams to Amazon EventBridge. All events that are delivered are valid. However, events are sent on a best-effort basis, which means there is no guarantee that:

- Events are delivered — A designated event can occur (e.g., a stream starts) but it is possible that Amazon IVS will not send a corresponding change event to EventBridge. Amazon IVS tries to deliver events for several hours before giving up.
- Events that are delivered will arrive in a specified timeframe — You may receive events up to a few hours old.
- Events that are delivered are ordered correctly — Events may be out of order, especially if they are sent within a short time of each other. For example, you could see stream down before stream up.

While it’s rare for events to be missing, late, or out of sequence, you should handle these possibilities if you write business-critical programs that depend on the order or existence of notification events.

You can create EventBridge rules for any of the following events.

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Event</th>
<th>Sent When ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVS Stream State Change</td>
<td>Stream Start</td>
<td>A stream is being processed and segments are available for the viewer to watch.</td>
</tr>
<tr>
<td>IVS Stream State Change</td>
<td>Stream End</td>
<td>A stream stops being processed and no longer produces segments for the viewer.</td>
</tr>
<tr>
<td>IVS Stream State Change</td>
<td>Stream Failure</td>
<td>A stream is not being processed and is not available because processing capacity was exceeded.</td>
</tr>
<tr>
<td>IVS Stream Health Change</td>
<td>Starvation Start</td>
<td>A stream is not receiving data from the streamer; the stream is said to be in “starvation.”</td>
</tr>
<tr>
<td>IVS Stream Health Change</td>
<td>Starvation End</td>
<td>A starving stream begins receiving data from the streamer and the stream is healthy again.</td>
</tr>
<tr>
<td>IVS Limit Breach</td>
<td>Ingest Bitrate</td>
<td>The incoming stream's bitrate exceeds the Amazon IVS limit.</td>
</tr>
<tr>
<td>IVS Limit Breach</td>
<td>Ingest Resolution</td>
<td>The incoming stream's resolution exceeds the Amazon IVS limit.</td>
</tr>
<tr>
<td>IVS Limit Breach</td>
<td>Concurrent Broadcasts</td>
<td>The total number of channels streaming at the same time exceeds the Amazon IVS limit.</td>
</tr>
</tbody>
</table>
### Event Type

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Event</th>
<th>Sent When ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVS Limit Breach</td>
<td>Concurrent Viewers</td>
<td>The total number of viewers watching your channels at the same time exceeds the Amazon IVS limit.</td>
</tr>
<tr>
<td>IVS Recording State Change</td>
<td>Recording Start</td>
<td>A stream starts being processed and segments are being written to the storage location configured for the channel.</td>
</tr>
<tr>
<td>IVS Recording State Change</td>
<td>Recording End</td>
<td>A stream ends and recording stops for this channel.</td>
</tr>
<tr>
<td>IVS Recording State Change</td>
<td>Recording Start Failure</td>
<td>A stream starts but recording fails to start due to errors (for example, the S3 bucket does not exist or is not in the correct region). This live stream is not recorded.</td>
</tr>
<tr>
<td>IVS Recording State Change</td>
<td>Recording End Failure</td>
<td>Recording ends with failure, due to errors encountered during recording. Some objects may still be written to the configured storage location.</td>
</tr>
</tbody>
</table>

**Note on stream IDs:** The `stream_id` field (in many events) is a unique stream identifier assigned each time a channel goes live. For a given channel, each live stream has a new `stream_id`. Hence, each channel ARN can have many corresponding stream IDs. Stream IDs allow customers to distinguish different stream sessions on the same channel.

**Note on latency of some events:** Encoder-configuration settings, especially the IDR/keyframe interval, affect the timing of stream startup and the latency of related events (Stream Start and Recording Start). A shorter keyframe interval decreases this latency. See "Reducing Latency" (p. 115) in *Amazon IVS Streaming Configuration* for information on setting IDR/Keyframe.

### Creating Amazon EventBridge Rules for Amazon IVS

You can create a rule that triggers on an event emitted by Amazon IVS. Follow the steps in Creating an EventBridge Rule That Triggers on an Event from an AWS Resource in the *Amazon EventBridge User Guide*. When you need to select a **Service name**, choose Interactive Video Service (IVS).

### Examples: Stream State Change

**Stream Start:** This event is sent when a stream is being processed and segments are available for the viewer.

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-012345678901",
    "detail-type": "IVS Stream State Change",
    "source": "aws.ivs",
    "account": "aws_account_id",
```
Stream End: This event is sent when a stream stops being processed and no longer produces segments for the viewer.

{  "version": "0",  "id": "01234567-0123-0123-0123-012345678901",  "detail-type": "IVS Stream State Change",  "source": "aws.ivs",  "account": "aws_account_id",  "time": "2017-06-12T10:23:43Z",  "region": "us-east-1",  "resources": [  "arn:aws:ivs:us-east-1:aws_account_id:channel/12345678-1a23-4567-a1bc-1a2b34567890"  ],  "detail": {  "event_name": "Stream End",  "channel_name": "Your Channel",  "stream_id": "st-1A2b3c4D5e6F78ghij9Klmn"  }}

Stream Failure: This event is sent when a stream is not being processed and is not available because processing capacity was exceeded.

{  "version": "0",  "id": "01234567-0123-0123-0123-012345678901",  "detail-type": "IVS Stream State Change",  "source": "aws.ivs",  "account": "aws_account_id",  "time": "2017-06-12T10:23:43Z",  "region": "us-east-1",  "resources": [  "arn:aws:ivs:us-east-1:aws_account_id:channel/12345678-1a23-4567-a1bc-1a2b34567890"  ],  "detail": {  "event_name": "Stream Failure",  "channel_name": "Your Channel",  "stream_id": "st-1A2b3c4D5e6F78ghij9Klmn",  "reason": "Transcode capacity exceeded. Please try again."  }}

Examples: Stream Health Change

Starvation Start: This event is sent when a stream is not receiving data from the streamer; the stream is said to be in "starvation."
Examples: Limit Breach

All limit-breath events include the name of the limit that is breached, the value of the limit, and the number by which the limit was exceeded (value at breach subtracted by the limit).

**Ingest Bitrate:** This event is sent when the incoming stream's bitrate exceeds the Amazon IVS limit.

```json
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "IVS Limit Breach",
  "source": "aws.ivs",
  "account": "aws_account_id",
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:channel/12345678-1a23-4567-a1bc-1a2b34567890"
  ],
  "detail": {
    "limit_name": "Ingest Bitrate",
    "limit_value": 1234,
    "breach_value": 2345
  }
}
```
Ingest Resolution: This event is sent when the incoming stream's resolution (total pixels or pixels per edge) exceeds the Amazon IVS limits.

Maximum total pixels exceeded:

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "IVS Limit Breach",
  "source": "aws.ivs",
  "account": "aws_account_id",
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:channel/12345678-1a23-4567-a1bc-1a2b34567890"
  ],
  "detail": {
    "limit_name": "Ingest Resolution",
    "limit_value": 495000,
    "exceeded_by": 426600,
    "limit_unit": "total pixels",
    "channel_name": "Your Channel",
    "stream_id": "st-1A2b3c4D5e6F78ghij9Klmn"
  }
}
```

Maximum pixels per edge exceeded:

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "IVS Limit Breach",
  "source": "aws.ivs",
  "account": "aws_account_id",
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:channel/12345678-1a23-4567-a1bc-1a2b34567890"
  ],
  "detail": {
    "limit_name": "Ingest Resolution",
    "limit_value": 855,
    "exceeded_by": 45,
    "limit_unit": "pixels per edge",
    "channel_name": "Your Channel",
    "stream_id": "st-1A2b3c4D5e6F78ghij9Klmn"
  }
}
```

Concurrent Broadcasts: This event is sent when the total number of channels streaming at the same time exceeds the Amazon IVS limit.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "IVS Limit Breach",
  "source": "aws.ivs",
  "account": "aws_account_id",
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:channel/12345678-1a23-4567-a1bc-1a2b34567890"
  ],
  "detail": {
    "limit_name": "Concurrent Broadcasts",
    "limit_value": 20,
    "exceeded_by": 21,
    "limit_unit": "channels",
    "channel_name": "Your Channel",
    "stream_id": "st-1A2b3c4D5e6F78ghij9Klmn"
  }
}
```
Concurrent Viewers: This event is sent when the total number of viewers watching your channels at the same time exceeds the Amazon IVS limit.

Examples: Recording State Change

For all recording state change events, the top-level path where all objects for this live stream are stored is `recording_s3_key_prefix`. In the case of failures, the reason for the failure is in `recording_status_reason`. The `recording_duration_ms` field is the number of milliseconds of recording duration.

Recording Start: This event is sent when a stream starts being processed and segments are being written to the storage location configured for the channel.
Examples: Recording State Change

```
{  
"version": "0",  
"id": "2020/eb93489721",  
"detail-type": "IVS Recording State Change",  
"source": "aws.ivs",  
"account": "123456789012",  
"time": "2020-06-24T07:51:32Z",  
"region": "us-west-2",  
"resources": [  
  "arn:aws:ivs:us-west-2:123456789012:channel/AbCdef1G2hij"  
],  
"detail": {  
  "channel_name": "Your Channel",  
  "stream_id": "st-1A2b3c4D5e6F78ghij9Klmn",  
  "recording_status": "Recording End",  
  "recording_status_reason": "",  
  "recording_s3_bucket_name": "r2s3-dev-channel-1-recordings",  
  "recording_s3_key_prefix": "ivs/v1/123456789012/AbCdef1G2hij/2020/6/23/20/12/j%20901ndcVs",  
  "recording_duration_ms": 0  
}  
}
```

**Recording End:** This event is sent when a stream ends and recording stops for this channel.

```

```
```

**Recording Start Failure:** This event is sent when a stream starts but recording fails to start due to errors (for example, the S3 bucket does not exist or is not in the correct region). This live stream is not recorded.

```
{  
"version": "0",  
"id": "2020/eb93489721",  
"detail-type": "IVS Recording State Change",  
"source": "aws.ivs",  
"account": "123456789012",  
"time": "2020-06-23T20:12:36Z",  
"region": "us-west-2",  
"resources": [  
  "arn:aws:ivs:us-west-2:123456789012:channel/AbCdef1G2hij"  
],  
"detail": {  
  "channel_name": "Your Channel",  
  "stream_id": "st-1A2b3c4D5e6F78ghij9Klmn",  
  "recording_status": "Recording Start Failure",  
  "recording_status_reason": "ValidationException",  
  "recording_s3_bucket_name": "r2s3-dev-channel-1-recordings",  
  "recording_s3_key_prefix": "",  
  "recording_duration_ms": 0  
}  
}
```

**Recording End Failure:** This event is sent when recording ends with failure, due to errors encountered during recording. Some objects may still be written to the configured storage location.

```
{  
"version": "0",  
"id": "2020/eb93489721",  
"detail-type": "IVS Recording State Change",  
"source": "aws.ivs",  
"account": "123456789012",  
"time": "2020-06-23T20:12:36Z",  
"region": "us-west-2",  
"resources": [  
  "arn:aws:ivs:us-west-2:123456789012:channel/AbCdef1G2hij"  
],  
"detail": {  
  "channel_name": "Your Channel",  
  "stream_id": "st-1A2b3c4D5e6F78ghij9Klmn",  
  "recording_status": "Recording End Failure",  
  "recording_status_reason": "",  
  "recording_s3_bucket_name": "r2s3-dev-channel-1-recordings",  
  "recording_s3_key_prefix": "",  
  "recording_duration_ms": 0  
}  
}
```
"version": "0",
"id": "12345678-1a23-4567-albc-1a2b34567890",
"detail-type": "IVS Recording State Change",
"source": "aws.ivs",
"account": "123456789012",
"time": "2020-06-24T07:51:32Z",
"region": "us-west-2",
"resources": [
  "arn:aws:ivs:us-west-2:123456a7-ab1c-2d34-e5f6-1a2b3c4d5678"
],
"detail": {
  "channel_name": "Your Channel",
  "stream_id": "st-1A2b3c4D5e6F78ghij9Klmn",
  "recording_status": "Recording End Failure",
  "recording_status_reason": "InternalServerException",
  "recording_s3_bucket_name": "r2s3-dev-channel-1-recordings",
  "recording_s3_key_prefix": "ivs/v1/123456789012/AbCdef1G2hij/2020/6/23/20/12/j8Z9091ndcVa",
  "recording_duration_ms": 0
}
}
Monitoring Amazon IVS with Amazon CloudWatch

You can monitor Amazon Interactive Video Service (IVS) resources using Amazon CloudWatch. CloudWatch collects and processes raw data from Amazon IVS into readable, near real-time metrics. These statistics are kept for 15 months, so you can gain a historical perspective on how your web application or service performs. You can set alarms for certain thresholds and send notifications or take actions when those thresholds are met. For details, see the CloudWatch User Guide.

The timestamp on a metric represents the start of the period during which metric data is accumulated. For example, suppose you get a per-minute `LiveDeliveredTime` metric sum of 300 seconds at 01:02:00. This would mean that 5 minutes' worth of video was served to viewers during the 1-minute period from 01:02:00 to 01:02:59.

Usually, metric data is emitted within 1 minute of the timestamp to which it refers.

For current information on data retention, search for "retention period" in Amazon CloudWatch FAQs.

Console Instructions

To access Amazon IVS metrics using the CloudWatch console:

2. In the navigation pane, choose Metrics.
3. On the All metrics tab, using the unlabeled dropdown at the left, select your "home" region, where your channel(s) was(were) created. For more on regions, see Global Solution, Regional Control (p. 1). For a list of supported regions, see the Amazon IVS page in the AWS General Reference.
4. If IVS appears as a selectable service under AWS Namespaces, select it. It will be listed if you use Amazon IVS and it is sending metrics to Amazon CloudWatch. (If IVS is not listed, you do not have any Amazon IVS metrics.) For example:
5. Choose a metric dimension as desired. For available dimensions, see the Metrics (p. 95) table below.

**CLI Instructions**

You also can access the metrics using the AWS CLI. This requires that you first download and configure the CLI on your machine. For details, see the AWS Command Line Interface User Guide.

Then, to access Amazon IVS metrics using the AWS CLI:

- At a command prompt, run:

  ```bash
  aws cloudwatch list-metrics --namespace AWS/IVS
  ```

  For more information, see Using Amazon CloudWatch Metrics in the Amazon CloudWatch User Guide.
# Metrics

Amazon IVS provides the following metrics in the **AWS/IVS** namespace.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
</table>
| ConcurrentViews   | None      | A count of concurrent views across all your live channels. A **view** is a unique viewing session which is actively downloading or playing video. *(For a more detailed definition, see the Glossary (p. 124).)* If channels are live but in aggregate have no views, the value of this metric is 0. If no channels are live, the metric has no data points. Unit: Count  
Valid statistics: Average, Maximum, Minimum — Average number, largest number, or smallest number (respectively) of concurrent views over the configured interval. |
| ConcurrentViews   | Channel   | Filters **ConcurrentViews** by channel ARN. If a channel is live but has no views, the value of this metric is 0. If a channel is not live, the metric has no data points.  
This metric provides data for a channel, not a stream. To see concurrent views for a particular streaming session on a given channel, evaluate the **ConcurrentViews** metric for that channel between the start and end times of the streaming session.  
Unit: Count  
Valid statistics: Average, Maximum, Minimum — Average number, largest number, or smallest number (respectively) of concurrent views over the configured interval. |
| ConcurrentStreams | None      | A count of your channels which are streaming live. If no channels are live, this metric has no data points.  
Unit: Count  
Valid statistics: Average, Maximum, Minimum — Average number, largest number, or smallest number (respectively) of concurrent streams over the configured interval. |
| LiveDeliveredTime | None      | Total real-time duration of video served to all viewers.  
Unit: Seconds  
Valid statistic: Sum |
<table>
<thead>
<tr>
<th>Metric</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LiveDeliveredTime</td>
<td>Channel</td>
<td>Filters LiveDeliveredTime by channel. Channel values are ARNs. Unit: Seconds</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid statistic: Sum</td>
</tr>
<tr>
<td>LiveDeliveredTime</td>
<td>Channel, ViewerCountryCode</td>
<td>Filters LiveDeliveredTime by channel and viewer's country code. Channel values are ARNs. Country values are two-character ISO 3166-1 country codes. This allows you to answer the question: where are my viewers watching from? If the viewer's country cannot be determined, it is shown as UNKNOWN. Unit: Seconds Valid statistic: Sum</td>
</tr>
<tr>
<td>LiveInputTime</td>
<td>None</td>
<td>Real-time duration of video stream. Unit: Seconds Valid statistic: Sum</td>
</tr>
<tr>
<td>LiveInputTime</td>
<td>Channel</td>
<td>Filters LiveInputTime by channel. Channel values are ARNs. Unit: Seconds</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid statistic: Sum</td>
</tr>
<tr>
<td>RecordedTime</td>
<td>None</td>
<td>Real-time duration of recorded video. Unit: Seconds Valid statistic: Sum</td>
</tr>
<tr>
<td>RecordedTime</td>
<td>Channel</td>
<td>Filters RecordedTime by channel. Channel values are ARNs. Unit: Seconds Valid statistic: Sum</td>
</tr>
</tbody>
</table>
Logging Amazon IVS API Calls with AWS CloudTrail

Amazon Interactive Video Service (IVS) is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or AWS service in Amazon IVS. CloudTrail captures all API calls for Amazon IVS as events. The calls captured include API calls from the Amazon IVS console and from your applications.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including Amazon IVS events. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in Event history. Using the information collected by CloudTrail, you can determine the request that was made to Amazon IVS, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Amazon IVS Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon IVS, that activity is recorded in a CloudTrail event along with other AWS service events in Event history. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon IVS, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the CloudTrail console, the trail applies to all AWS regions. The trail logs events from all Regions in the AWS partitions and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to analyze and act on the event data collected in CloudTrail logs. For more information, see these items in the CloudTrail User Guide:

- Creating a Trail For Your AWS Account (overview)
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions
- Receiving CloudTrail Log Files from Multiple Accounts

All Amazon IVS actions are logged by CloudTrail and documented in the Amazon IVS API Reference. For example, calls to the CreateChannel, ListChannels, and DeleteChannel endpoints generate entries in the CloudTrail log files.

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

- With root or AWS Identity and Access Management (IAM) user credentials
- With temporary security credentials for a role or federated user.
- By another AWS service.

For more information, see the CloudTrail userIdentity Element.
Understanding Amazon IVS Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on.

CloudTrail log files contain one or more log entries. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example shows a CloudTrail log entry for the CreateChannel endpoint.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:assumed-role/FirstStreamer/1234567890123456789",
        "accountId": "123456789012",
        "accessKeyId": "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "FirstStreamer"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-04-02T20:57:43Z"
            }
        },
        "eventTime": "2020-04-02T20:57:46Z",
        "eventSource": "ivs.amazonaws.com",
        "eventName": "CreateChannel",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "10.10.10.10",
        "userAgent": "console.amazonaws.com",
        "requestParameters": {
            "name": "default"
        },
        "responseElements": {
            "channel": {
                "arn": "arn:aws:ivs:us-west-2:123456789012:channel/1EXAMPLE",
                "authorized": false,
                "ingestEndpoint": "EXAMPLE.global-contribute.live-video.net",
                "latencyMode": "LOW",
                "name": "default",
                "playbackUrl": "https://EXAMPLE.m3u8",
                "tags": {}
            },
            "streamKey": {
                "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/1EXAMPLE",
                "tags": {}
            }
        },
        "requestID": "1234567890123456789",
        "eventType": "AwsApiCall",
        "requestAttributes": {}
    }
}
```
"recipientAccountId": "123456789012"}
Amazon IVS Security

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** — AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs.
- **Security in the cloud** — Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your organization’s requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon IVS. The following topics show you how to configure Amazon IVS to meet your security and compliance objectives.

**Topics**
- Data Protection (p. 100)
- Identity and Access Management (p. 101)
- Using Service-Linked Roles for Amazon IVS (p. 107)
- Logging and Monitoring (p. 109)
- Incident Response (p. 109)
- Resilience (p. 109)
- Infrastructure Security (p. 110)

### Data Protection

For data sent to Amazon Interactive Video Service (IVS), the following data protections are in place:

- Amazon IVS encrypts data in transit via HTTPS API endpoints, RTMPS ingest, and HTTPS playback. No configuration is required for the API endpoints.
- For ingest, streamers can secure their content by using RTMPS. This is available by default. See Getting Started with Amazon IVS (p. 3).
- For transcoding/transmuxing, data may be transmitted unencrypted on internal Amazon networks.
- For playback, data is served over HTTPS.
- Live-video content is not stored and is ephemeral. It simply travels through the system and is cached (on internal systems) while being viewed.
- For the auto-record-to-S3 feature, video content is written to Amazon S3. For more information, see data protection in Amazon S3.
- All stored, customer-input metadata is in AWS-managed services using server-side encryption.
• To improve quality of service, Amazon IVS stores customer (end user) metadata (for example, buffer rates for a particular region). This metadata cannot be used to personally identify your end users.
• Public encryption keys (which you manage) can be used with the ImportPlaybackKeyPair API endpoint. See the Amazon IVS API Reference. Do not share these encryption keys.

Amazon IVS does not require that you supply any customer (end user) data. There are no fields in channels, inputs, or input security groups where there is an expectation that you will provide customer (end user) data.

Do not put sensitive identifying information such as your customer (end user) account numbers into free-form fields such as a Name field. This includes when you work with the Amazon IVS console or API, AWS CLI, or AWS SDKs. Any piece of data that you enter into Amazon IVS might be included in diagnostic logs.

Streams are not end-to-end encrypted; a stream may be transmitted unencrypted internally in the IVS network, for processing.

Identity and Access Management

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Amazon IVS resources. IAM is a feature of your AWS account offered at no additional charge.

Important: For comprehensive information, see the AWS IAM product page and Signing AWS API Requests. The product page contains many additional links to AWS IAM documentation, including the IAM User Guide. Throughout this section, we also provide links to specific sections of the IAM User Guide. You should be familiar with this material before proceeding.

Audience

How you use IAM differs, depending on the work you do in Amazon IVS:

• Service user – If you use the Amazon IVS service to do your job, your administrator provides you with the credentials and permissions that you need. As you use more Amazon IVS features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon IVS, see Troubleshooting (p. 106).
• Service administrator – If you’re in charge of Amazon IVS resources at your company, you probably have full access to Amazon IVS. It’s your job to determine which Amazon IVS features and resources your employees should access. You must then submit requests to your IAM administrator, to change the permissions of your service users. Review the information on this page to understand basic IAM concepts. To learn more about how your company can use IAM with Amazon IVS, see How Amazon IVS Works with IAM (p. 101).
• IAM administrator – If you’re an IAM administrator, you can write policies to manage access to Amazon IVS. To view example Amazon IVS identity-based policies that you can use in IAM, see Identity-Based Policy Examples (p. 103).

How Amazon IVS Works with IAM

Before you can make Amazon IVS API requests, you must create one or more IAM identities (users, groups, and roles) and IAM policies, then attach policies to identities. It takes up to a few minutes for the permissions to propagate; until then, API requests are rejected.
For a high-level view of how Amazon IVS works with IAM, see AWS Services That Work with IAM in the IAM User Guide.

Identities

You can create IAM identities to provide authentication for people and processes in your AWS account. IAM groups are collections of IAM users that you can manage as a unit. See Identities (Users, Groups, and Roles) in the IAM User Guide.

Policies

See these sections in the IAM User Guide:

- Access Management — All about policies.
- Actions, Resources, and Condition Keys for Amazon IVS
- AWS Global Condition Context Keys
- IAM JSON Policy Elements Reference — All the elements that you can use in a JSON policy.

By default, IAM users and roles don't have permission to create or modify Amazon IVS resources (even to change their own passwords). They also cannot perform tasks using the AWS console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources that they need.

IAM policies define permissions for an action regardless of the method that is used to perform the operation. For example, suppose that you have a policy that allows the iam:GetRole action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Policies are JSON permissions-policy documents made up of elements. Amazon IVS supports three elements:

- **Actions** — Policy actions for Amazon IVS use the ivs prefix before the action. For example, to grant someone permission to create an Amazon IVS channel with the Amazon IVS CreateChannel API method, you include the ivs:CreateChannel action in the policy for that person. Policy statements must include either an Action or NotAction element.

- **Resources** — The Amazon IVS channel resource has the following ARN format:

  arn:#{Partition}:ivs:#{Region}:#{Account}:channel/#{channelId}

  For example, to specify the VgNkEJgOVX9N channel in your statement, use this ARN:

  "Resource": "arn:aws:ivs:us-west-2:123456789012:channel/VgNkEJgOVX9N"

  Some Amazon IVS actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (*):

  "Resource":"**"

- **Conditions** — Amazon IVS supports some global condition keys: aws:RequestTag, aws:TagKeys, and aws:ResourceTag.

You can use variables as placeholders in a policy. For example, you can grant an IAM user permission to access a resource only if it is tagged with the user's IAM username. See Variables and Tags in the IAM User Guide.
Authorization Based on Amazon IVS Tags

You can attach tags to Amazon IVS resources or pass tags in a request to Amazon IVS. To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys. For more information about tagging Amazon IVS resources, see “Tagging” in the Amazon IVS API Reference.

For an example, see View Amazon IVS Channels Based on Tags (p. 106).

Roles

See IAM Roles and Temporary Security Credentials in the IAM User Guide.

An IAM role is an entity within your AWS account that has specific permissions.

Amazon IVS supports using temporary security credentials. You can use temporary credentials to sign in with federation, assume an IAM role, or assume a cross-account role. You obtain temporary security credentials by calling AWS Security Token Service API operations such as AssumeRole or GetFederationToken.

Privileged and Unprivileged Access

API resources have privileged access. Unprivileged playback access can be set up through private channels; see Setting Up Private Channels (p. 69).

Best Practices for Policies

See IAM Best Practices in the IAM User Guide.

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Amazon IVS resources in your account. These actions can incur costs for your AWS account. Follow these recommendations:

- **Grant least privilege** — When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant more permissions as needed. Doing so is more secure than starting with permissions that are too lenient, then trying to tighten them later. Specifically, reserve `ivs:*` for admin access; do not use it in applications.

- **Enable multi-factor authentication (MFA) for sensitive operations** — For extra security, require IAM users to use MFA to access sensitive resources or API operations.

- **Use policy conditions for extra security** — To the extent practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses from which a request must come. You also can write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA.

Identity-Based Policy Examples

Use the Amazon IVS Console

To access the Amazon IVS console, you must have a minimum set of permissions which allow you to list and view details about the Amazon IVS resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console will not function as
intended for identities with that policy. To ensure access to the Amazon IVS console, attach the following policy to the identities (see Adding and Removing IAM Permissions in the IAM User Guide).

The four parts of the following policy provide access to:

- All Amazon IVS API endpoints
- Your Amazon Amazon IVS service quotas (p. 111)
- Amazon S3 endpoints needed for the Amazon IVS auto-record-to-S3 functionality (creating a new S3 bucket, choosing an existing bucket, filtering buckets for current region)
- Auto-record-to-S3 service-linked-role creation

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ivs:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "servicequotas:ListServiceQuotas"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:AttachRolePolicy",
        "iam:PutRolePolicy"
      ],
      "Resource": "arn:aws:iam::*:role/aws-service-role/ivs.amazonaws.com/AWSServiceRoleForIVSRecordToS3*"
    }
  ]
}
```

Allow Users to View Their Own Permissions

This example shows a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the AWS console or programmatically using the AWS CLI or AWS API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:GetUserPolicyVersion",
        "iam:GetUserPolicyAttachedToRolePolicy",
        "iam:GetUserPolicyAttachedToRole"
      ],
      "Resource": "*"
    }
  ]
}
```
Identity-Based Policy Examples

"Effect": "Allow",
"Action": [
   "iam:GetUserPolicy",
   "iam:ListGroupsForUser",
   "iam:ListAttachedUserPolicies",
   "iam:ListUserPolicies",
   "iam:GetUser"
],
"Resource": [
   "arn:aws:iam:*:*:user/${aws:username}"
]
},

{"Sid": "NavigateInConsole",
 "Effect": "Allow",
 "Action": [
    "iam:GetGroupPolicy",
    "iam:GetPolicyVersion",
    "iam:GetPolicy",
    "iam:ListAttachedGroupPolicies",
    "iam:ListGroupPolicies",
    "iam:ListPolicyVersions",
    "iam:ListPolicies",
    "iam:ListUsers"
],
"Resource": "*
"
}

Access an Amazon IVS Channel

Here, you want to grant an IAM user in your AWS account access to one of your Amazon IVS channels, VgNkEJgOVX9N. You also want to allow the user to stop the stream (ivs:StopStream), add metadata (ivs:PutMetadata), and update the channel (ivs:UpdateChannel). The policy also grants permissions required by the Amazon IVS console: ivs:ListChannels, ivs:ListStreams, ivs:GetChannel, and ivs:GetStream.

{
 "Version":"2012-10-17",
 "Statement":[
    {
       "Sid":"ListChannelsInConsole",
       "Effect": "Allow",
       "Action": [
           "ivs:ListChannels",
           "ivs:ListStreams"
       ],
       "Resource": "arn:aws:ivs:*:*:channel/*"
    },
    {
       "Sid":"ViewSpecificChannelInfo",
       "Effect": "Allow",
       "Action": [
           "ivs:GetChannel",
           "ivs:GetStream"
       ],
       "Resource": "arn:aws:ivs:*:*:channel/VgNkEJgOVX9N"
    },
    {
       "Sid":"ManageChannel",
       "Effect": "Allow",
       "Action":
    }]
}
View Amazon IVS Channels Based on Tags

You can use conditions in your identity-based policy to control access to Amazon IVS resources based on tags. This example shows a policy that allows viewing a channel. This policy also grants the permissions necessary to complete this action on the Amazon IVS console.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListWidgetsInConsole",
            "Effect": "Allow",
            "Action": "ivs:ListChannels",
            "Resource": "arn:aws:ivs:*:*:channel/*"
        },
        {
            "Sid": "ViewChannelIfOwner",
            "Effect": "Allow",
            "Action": "ivs:GetChannel",
            "Resource": "arn:aws:ivs:*:*:channel/*",
            "Condition": {
                "StringEquals": "aws:ResourceTag/Owner": "+{aws:username}"
            }
        }
    ]
}
```

You can attach this policy to the IAM users in your account. However, permission is granted only if the channel is tagged with that user's username as an owner. If a user named richard-roe tries to view an Amazon IVS channel, the channel must be tagged Owner=richard-roe or owner=richard-roe; otherwise he is denied access. (The condition tag key Owner matches both Owner and owner because condition-key names are not case sensitive.)

Troubleshooting

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

- **I am not authorized to perform an action in Amazon IVS.**

  The following example error occurs when the mateojackson IAM user tries to use the AWS console to view details about a channel but does not have ivs:GetChannel permission.

  ```text
  User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
  ivs:GetChannel on resource: arn:aws:ivs:us-west-2:123456789012:channel/VgNkEJgOVX9N
  ```

  In this case, Mateo asks his administrator to update his policies to allow him to access the arn:aws:ivs:us-west-2:123456789012:channel/VgNkEJgOVX9N resource using the ivs:GetChannel action.
• I want to view my access keys.

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair. Access keys have two parts:

• An access key ID (for example, AKIAIOSFODNN7EXAMPLE)
• A secret access key (for example, wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY)

As with a username and password, you must use both the access key ID and the secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

_Important: Do not give your access keys to a third party, even to help find your canonical user ID. Doing so might give someone permanent access to your account._

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only when you create it. If you lose your secret access key, you must add new access keys to your IAM user.

You can have at most two access keys. If you already have two, you must delete one key pair before creating a new one. See Managing Access Keys for IAM Users in the IAM User Guide.

• I’m an administrator and want to allow others to access Amazon IVS.

To allow others to access Amazon IVS, you must create an IAM entity (user or role) for the person or application that needs access. The person or application will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants the correct permissions in Amazon IVS.

To get started, see Creating Your First IAM Delegated User and Group in the IAM User Guide.

• I want to allow people outside my AWS account to access my Amazon IVS resources.

You can create a role that users in other accounts or people outside your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources. For related information, see these sections of the IAM User Guide:

<table>
<thead>
<tr>
<th>To learn ...</th>
<th>See ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to provide access to your resources across AWS accounts</td>
<td>Providing Access to an IAM User in Another AWS Account That You Own</td>
</tr>
<tr>
<td>that you own</td>
<td></td>
</tr>
<tr>
<td>How to provide access to your resources to third-party AWS</td>
<td>Providing Access to AWS Accounts Owned by Third Parties</td>
</tr>
<tr>
<td>accounts</td>
<td></td>
</tr>
<tr>
<td>How to provide access through identity federation</td>
<td>Providing Access to Externally Authenticated Users (Identity Federation)</td>
</tr>
<tr>
<td>The difference between using roles and resource-based policies</td>
<td>How IAM Roles Differ from Resource-based Policies</td>
</tr>
<tr>
<td>for cross-account access</td>
<td></td>
</tr>
</tbody>
</table>

Using Service-Linked Roles for Amazon IVS

Amazon IVS uses IAM service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Amazon IVS. Service-linked roles are predefined by Amazon IVS and include all the permissions that the service requires to call other AWS services on your behalf.
A service-linked role makes setting up Amazon IVS easier because you don't have to manually add the necessary permissions. Amazon IVS defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon IVS can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting its related resources. This protects your Amazon IVS resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have Yes in the Service-Linked Role column. Choose a Yes with a link to view the service-linked role documentation for that service.

### Service-Linked Role Permissions for Amazon IVS

Amazon IVS uses the service-linked role named **AWSServiceRoleForIVSRecordToS3** to access Amazon S3 buckets on behalf of your Amazon IVS Channels.

The AWSServiceRoleForIVSRecordToS3 service-linked role trusts the following services to assume the role:

- ivs.amazonaws.com

The role permissions policy allows Amazon IVS to complete the following actions on the specified resources:

- **Action:** s3:PutObject on your Amazon S3 buckets

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-Linked Role Permissions in the IAM User Guide.

### Creating a Service-Linked Role for Amazon IVS

You don't need to manually create a service-linked role. When you create a recording-configuration resource in the Amazon IVS Console, the AWS CLI, or the AWS API, Amazon IVS creates the service-linked role for you.

**Important**

This service-linked role can appear in your account if you completed an action in another service that uses the features supported by this role. To learn more, see A New Role Appeared in My IAM Account.

If you delete this service-linked role and then need to create it again, you can use the same process to recreate the role in your account. When you create a recording-configuration resource, Amazon IVS creates the service-linked role for you again.

### Editing a Service-Linked Role for Amazon IVS

Amazon IVS does not allow you to edit the AWSServiceRoleForIVSRecordToS3 service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a Service-Linked Role in the IAM User Guide.

### Deleting a Service-Linked Role for Amazon IVS

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored.
or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

**Note**
If the Amazon IVS service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

**To delete Amazon IVS resources used by the AWSServiceRoleForIVSRecordToS3 service-linked role:**

Use the Amazon IVS Console, the AWS CLI, or the AWS API to remove the recording-configuration association from all channels and delete all recording-configuration resources in the region.

**To manually delete the service-linked role using IAM:**

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForIVSRecordToS3 service-linked role. For more information, see Deleting a Service-Linked Role in the IAM User Guide.

### Supported Regions for Amazon IVS Service-Linked Roles

Amazon IVS supports using service-linked roles in all of the regions where the service is available. For more information, see Amazon IVS Service Endpoints.

### Logging and Monitoring

To log performance and/or operations, use Amazon CloudTrail. See Logging Amazon IVS API Calls with AWS CloudTrail (p. 97).

### Incident Response

To detect or alert for incidents, you can monitor your stream’s health via Amazon EventBridge events. See Using Amazon EventBridge with Amazon IVS (p. 85).

Use the AWS status page for information on the overall health of Amazon IVS.

### Resilience

The Amazon IVS API uses the AWS global infrastructure and is built around AWS Regions and Availability Zones. AWS Regions provide multiple Availability Zones, which are:

- Physically separated and isolated.
- Connected with low-latency, high-throughput, highly-redundant networking.
- More available, fault tolerant, and scalable than traditional single or multiple data-center infrastructures.

You can design and operate applications and databases that automatically fail over between Availability Zones to avoid any interruption. For more information on the API, see the Amazon IVS API Reference. For more information on AWS Regions and Availability Zones, see AWS Global Infrastructure.
Amazon IVS Video Data Plane

Video ingestion and distribution run over Amazon IVS’s Content Delivery Network (CDN). The CDN is specialized and highly tuned for low-latency video. This enables Amazon IVS to provide customers with end-to-end, high quality video served to a global audience with minimal delay. The video CDN has global Points-of-Presence (PoPs), allowing broadcasters and viewers to be geographically dispersed.

Regardless of the AWS region where you chose to configure your Amazon IVS resources:

- Streamers automatically ingest video to a PoP geographically close to their location.
- Viewers stream video via the global video CDN.

Once ingested, video streams are processed and transcoded in one of several Amazon IVS datacenters. Amazon IVS does not provide automated failover for ingestion or transcoding failures. Instead, streamers should configure their encoders or broadcasting clients to automatically re-ingest on any broadcasting failures.

Infrastructure Security

As a managed service, Amazon IVS is protected by the AWS global network security procedures. These are described in Best Practices for Security, Identity, & Compliance.

API Calls

You use AWS published API calls to access Amazon IVS through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later (due to vulnerabilities in earlier versions). Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Also, API requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service to generate temporary security credentials to sign requests.

You can call these API operations from any network location, but Amazon IVS does support resource-based access policies, which can include restrictions based on the source IP address. You can also use Amazon IVS policies to control access from specific Amazon Virtual Private Cloud (Amazon VPC) endpoints or specific VPCs. Effectively, this isolates network access to a given Amazon IVS resource from only the specific VPC within the AWS network.

Also, all API requests are signed sigv4.

For API details, see the Amazon IVS API Reference.

Streaming and Playback

Playback happens over HTTPS from the edge to the viewer, and the “contribution edge” (ingest endpoint) supports RTMPS (RTMP over TLS). Amazon IVS streaming requires TLS version 1.2 or later. Streams are not end-to-end encrypted; a stream may be transmitted unencrypted internally in the IVS network, for processing.
Amazon IVS Service Quotas

The following are service quotas and limits for Amazon Interactive Video Service (IVS) endpoints, resources, and other operations. Service quotas (also known as limits) are the maximum number of service resources or operations for your AWS account. That is, these limits are per AWS account, unless noted otherwise in the table. Also see AWS Service Quotas.

You use an endpoint to connect programmatically to an AWS service. Also see AWS Service Endpoints.

Important: All accounts have limits on the number of concurrent views and concurrent streams. (A view is a unique viewing session which is actively downloading or playing video. For a more detailed definition, see the Glossary (p. 124).) Ensure that your limits are adequate and request an increase if needed, especially if you are planning a large streaming event.

Service Quota Increases

For quotas that are adjustable, you can request a rate increase through the AWS console. Use the console to view information about service quotas too.

Only the following quotas are adjustable:

- Channels (see Resource Quotas (p. 112))
- Recording Configurations (see Resource Quotas (p. 112))
- Concurrent views (see Contribution and Playback Quotas (p. 112))
- Concurrent streams (see Contribution and Playback Quotas (p. 112))

API Call Rate Quotas

<table>
<thead>
<tr>
<th>Endpoint Type</th>
<th>Endpoint</th>
<th>Default Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel</td>
<td>BatchGetChannel</td>
<td>5 TPS</td>
</tr>
<tr>
<td>Channel</td>
<td>CreateChannel</td>
<td>5 TPS</td>
</tr>
<tr>
<td>Channel</td>
<td>DeleteChannel</td>
<td>5 TPS</td>
</tr>
<tr>
<td>Channel</td>
<td>GetChannel</td>
<td>5 TPS</td>
</tr>
<tr>
<td>Channel</td>
<td>ListChannels</td>
<td>5 TPS</td>
</tr>
<tr>
<td>Channel</td>
<td>UpdateChannel</td>
<td>5 TPS</td>
</tr>
<tr>
<td>Playback key pair</td>
<td>DeletePlaybackKeyPair</td>
<td>3 TPS</td>
</tr>
<tr>
<td>Playback key pair</td>
<td>GetPlaybackKeyPair</td>
<td>3 TPS</td>
</tr>
<tr>
<td>Playback key pair</td>
<td>ImportPlaybackKeyPair</td>
<td>3 TPS</td>
</tr>
<tr>
<td>Playback key pair</td>
<td>ListPlaybackKeyPairs</td>
<td>3 TPS</td>
</tr>
<tr>
<td>Recording configuration</td>
<td>CreateRecordingConfiguration</td>
<td>3 TPS</td>
</tr>
<tr>
<td>Recording configuration</td>
<td>DeleteRecordingConfiguration</td>
<td>3 TPS</td>
</tr>
</tbody>
</table>
Resource Quotas

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default Quota</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channels</td>
<td>5000</td>
<td>Maximum number of channels, per AWS region.</td>
</tr>
<tr>
<td>Playback authorization key pairs</td>
<td>3</td>
<td>Maximum number of playback authorization key pairs, per AWS region.</td>
</tr>
<tr>
<td>Recording configurations</td>
<td>20</td>
<td>Maximum number of recording configurations, per AWS region.</td>
</tr>
<tr>
<td>Stream key</td>
<td>1</td>
<td>Maximum number of stream keys, per channel.</td>
</tr>
</tbody>
</table>

Contribution and Playback Quotas

<table>
<thead>
<tr>
<th>Resource or Feature</th>
<th>Default Quota</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Playback token size</td>
<td>2 KB</td>
<td>Maximum size of the entire JSON web token (JWT) used to initiate playback.</td>
</tr>
</tbody>
</table>
### Service Quotas Integration with CloudWatch Usage Metrics

You can use CloudWatch to proactively manage your service quotas, via CloudWatch usage metrics. You can use these metrics to visualize your current service usage on CloudWatch graphs and dashboards. Amazon IVS usage metrics correspond to Amazon IVS service quotas.

You can use a CloudWatch metric math function to display the service quotas for those resources on your graphs. You can also configure alarms that alert you when your usage approaches a service quota.

To access usage metrics:

1. Open the Service Quotas console at https://console.aws.amazon.com/servicequotas/

<table>
<thead>
<tr>
<th>Resource or Feature</th>
<th>Default Quota</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrent views</td>
<td>15000</td>
<td>Maximum number of views allowed to play back a live channel, across all channels in an AWS region.</td>
</tr>
<tr>
<td>Metadata payload</td>
<td>1 KB</td>
<td>Maximum size of a <code>PutMetadata</code> request payload (Amazon IVS API).</td>
</tr>
<tr>
<td>Concurrent streams</td>
<td>100</td>
<td>Maximum number of channels that can be streamed simultaneously, per AWS region. If you exceed this threshold, the stream is rejected.</td>
</tr>
<tr>
<td>Ingest bitrate (if channel type is STANDARD)</td>
<td>8.5 Mbps</td>
<td>Maximum bits per second that can be streamed to a channel whose <code>type</code> is <code>STANDARD</code> (the default). Warning: If you exceed this threshold, the stream probably will disconnect immediately. See the Amazon IVS API Reference for details about channel type.</td>
</tr>
<tr>
<td>Ingest bitrate (if channel type is BASIC)</td>
<td>1.5 Mbps</td>
<td>Maximum bits per second that can be streamed to a channel whose <code>type</code> is <code>BASIC</code>. Warning: If you exceed this threshold, the stream probably will disconnect immediately. See the Amazon IVS API Reference for details about channel type.</td>
</tr>
<tr>
<td>Ingest resolution (if channel type is STANDARD)</td>
<td>1080p (2.1M total pixels, 1920 pixels/edge)</td>
<td>Maximum resolution in pixels that can be streamed to a channel whose <code>type</code> is <code>STANDARD</code> (the default). There are two relevant thresholds: total pixels and pixels per edge. Warning: If you exceed either of these thresholds, the stream probably will disconnect immediately. See the Amazon IVS API Reference for details about channel type.</td>
</tr>
<tr>
<td>Ingest resolution (if channel type is BASIC)</td>
<td>480p (415K total pixels, 855 pixels/edge)</td>
<td>Maximum resolution in pixels that can be streamed to a channel whose <code>type</code> is <code>BASIC</code>. There are two relevant thresholds: total pixels and pixels per edge. Warning: If you exceed either of these thresholds, the stream probably will disconnect immediately. See the Amazon IVS API Reference for details about channel type.</td>
</tr>
</tbody>
</table>
2. In the navigation pane, select **AWS services**.
3. From the AWS services list, search for and select **Amazon Interactive Video Service**.
4. In the **Service quotas** list, select the service quota of interest. A new page opens with information about the service quota/metric.

Alternately, you can get to these metrics through the CloudWatch console. Under **AWS Namespaces**, choose **Usage**. Then, from the **Service** list, choose **IVS**. (See Monitoring Amazon IVS with Amazon CloudWatch (p. 93).)

In the **AWS/Usage** namespace, Amazon IVS provides the following metric:

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourceCount</td>
<td>A count of the specified resources running in your account. The resources are defined by the dimensions associated with the metric. Valid statistic: Maximum (the maximum number of resources used during the 1-minute period).</td>
</tr>
</tbody>
</table>

The following dimensions are used to refine the usage metric:

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>The name of the AWS service containing the resource. Valid value: IVS.</td>
</tr>
<tr>
<td>Class</td>
<td>The class of resource being tracked. Valid value: None.</td>
</tr>
<tr>
<td>Type</td>
<td>The type of resource being tracked. Valid value: Resource.</td>
</tr>
<tr>
<td>Resource</td>
<td>The name of the AWS resource. Valid values: ConcurrentStreams, ConcurrentViews. The ConcurrentStreams and ConcurrentViews usage metrics are copies of the ones in the <strong>AWS/IVS</strong> namespace (with the None dimension), as described in Monitoring Amazon IVS with Amazon CloudWatch (p. 93).</td>
</tr>
</tbody>
</table>

**Creating a CloudWatch Alarm for Usage Metrics**

To create a CloudWatch alarm based on an Amazon IVS usage metric:

1. From the Service Quotas console, select the service quota of interest, as described above. Currently, alarms can be created only for ConcurrentStreams and ConcurrentViews.
2. In the **Amazon CloudWatch alarms** section, choose **Create**.
3. From the **Alarm threshold** dropdown list, choose the percentage of your applied quota value that you want to set as the alarm value.
4. For **Alarm name**, enter a name for the alarm.
5. Select **Create**.
Amazon IVS Streaming Configuration

Amazon Interactive Video Service (IVS) allows developers to easily deliver low-latency video to viewers worldwide. With Amazon IVS, streamers need to handle only stream production, then send the stream to Amazon IVS. Amazon IVS handles video processing (ingesting and transcoding), delivery, and playback to viewers using the Amazon IVS player.

There is a wealth of solutions for live streaming. Whether you have a studio equipped with multiple cameras, visual switchers, graphics compositing, and a variety of audio mixing equipment, or you plan to start your first stream off a smartphone, you need to deal with some of the same concepts and encoding parameters.

This document describes how to configure video encoders to stream to Amazon IVS. The audience for this document is developers who want to build streaming functionality into their applications.

Prerequisites

Follow the steps in Getting Started with Amazon IVS (p. 3), to create a channel and set up streaming. In the process, a channel ARN (Amazon Resource Name) and stream key are assigned, along with URLs for ingesting and playing back a stream. You will need to point your streaming application to the ingest URL.

Before reading this document, you should be familiar with:

- Amazon IVS basics: Read What is Amazon IVS (p. 1) and Getting Started with Amazon IVS (p. 3)
- Amazon IVS API: Understand the Amazon IVS API Reference.

Reducing Latency

Amazon IVS low-latency streaming is compatible with most streaming applications and requires only minor changes to your streaming-application configuration. For the lowest possible latency, you must use the Amazon IVS player; third-party HLS video players are not supported. See the Amazon IVS Player SDK documentation.

To prepare your streaming application for low-latency streaming, do the following. (Note: not all these options are available on every streaming application.)

- On the video encoder, set IDR/Keyframe to a 2-second interval (or 1 second, for even lower end-to-end latency).

  IDR/Keyframe directly affects the timing of stream startup and the latency of related EventBridge events (Stream Start and Recording Start). If IDR/Keyframe is 2 seconds, stream-start latency will be approximately 9-12 seconds. If IDR/Keyframe is 1 second, stream-start latency will be approximately 6-8 seconds. Your video will be available for viewers and auto-recording to Amazon S3 only after the initial stream-start latency period.

  The shorter, 1-second keyframe interval has some QoS tradeoffs. It can cause the Amazon IVS Player’s adaptive bitrate streaming (ABR) to switch resolution more often; the segment size is smaller, so the
ABR check happens more often. Buffering may increase due to increased resolution-switching and/or if the viewer’s network cannot download the segments fast enough. Evaluate these tradeoffs when deciding between a 1- or 2-second keyframe interval.

- If available, set your encoder to zero-latency tuning within an x264 configuration.
- Ensure that buffer size (VBV) does not exceed the average bitrate (kilobits-per-second) of the stream.

**Avoid Third-Party Streaming/Forwarding Services**

We strongly recommend you do not use third-party service to restream or forward content to Amazon IVS. *This will incur extra latency.* For low latency, stream directly to Amazon IVS.

**Encoder Settings**

**Stream Ingest: Codecs, RTMPS, and Port 443**

*Codecs:* Amazon IVS supports H.264 for video and AAC (LC) for audio.

Amazon IVS supports the most common secure ingest protocol used in streaming software and hardware, RTMPS (Real-Time Messaging Protocol over a TLS/SSL connection). Amazon IVS streaming and playback require TLS version 1.2 or later.

Your video encoder must connect to Amazon IVS ingest over the RTMPS protocol associated with outbound port 443/TCP. To ensure this, include the port in your ingest URL path; for example:

```
rtmps://<IVS-ingest-server>:443/app/<IVS-stream-key>
```

**Resolution/Bitrate/FPS**

The stream’s resolution largely determines its bitrate and frame rate (frames-per-second, or FPS). Use the following guidelines; these are our recommendations. Note the resolutions shown below are landscape orientation (horizontal x vertical), so reverse these for portrait orientation.

<table>
<thead>
<tr>
<th></th>
<th>Acceptable Quality (SD) 480p (852x480)</th>
<th>Good Quality (HD) 720p (1280x720)</th>
<th>High Quality (Full HD) 1080p (1920x1080)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bitrate</strong></td>
<td>Up to 1500 Kbps</td>
<td>Up to 4500 Kbps</td>
<td>Up to 8500 Kbps</td>
</tr>
<tr>
<td><strong>FPS</strong></td>
<td>30</td>
<td>30 or 60</td>
<td>30 or 60</td>
</tr>
<tr>
<td><strong>Keyframe interval</strong></td>
<td>2 seconds</td>
<td>2 seconds</td>
<td>2 seconds</td>
</tr>
</tbody>
</table>

Bitrate, FPS, and resolution are interrelated. The optimal values depend on circumstances and can be complicated to determine. Our best guidance is to start with the values above and experiment if desired. The goal is clear and smooth motion of video components during streaming and good resolution within the available bandwidth. Increasing frame rate and/or resolution increases overall video quality, but this is necessarily limited by bandwidth.

Amazon IVS supports framerates up to 60 FPS. The higher the framerate, the better the quality -- as long as there is adequate bitrate bandwidth. Depending on the application, a low framerate can be fine; e.g., for a security camera.
Channel Types

Channel type determines the allowable resolution and bitrate. *If you exceed the allowable resolution or bitrate, the stream probably will disconnect immediately.* Amazon IVS supports two channel types:

- **STANDARD:** Multiple qualities are generated from the original input, to automatically give viewers the best experience for their devices and network conditions. Resolution can be up to 1080p and bitrate can be up to 8.5 Mbps. Audio is transcoded only for renditions 360p and below; above that, audio is passed through. This is the default.
- **BASIC:** Amazon IVS delivers the original input to viewers. The viewer’s video-quality choice is limited to the original input. Resolution can be up to 480p and bitrate can be up to 1.5 Mbps.

Video Settings

We recommend the following settings. They are available to most H.264 video-encoding software or hardware APIs.

- On the video encoder, set IDR/Keyframe to a 2-second interval (or 1 second, for even lower end-to-end latency).
- H.264 level: Main
- Scene change: Off (preferred)
- Chroma subsample: YUV420P
- CABAC: Preferred

Audio Settings

We support the following settings:

- Codec: AAC (LC)
- Bitrate: 96 Kbps to 320 Kbps
- Sample rate: 44.1 Khz or 48 Khz (it is best to match your production audio flow)
- Channels: Maximum 2 - Stereo (1: mono or 2: stereo audio channel support)

Use CBR, Not VBR

Always use CBR (Constant BitRate), not VBR (Variable BitRate), as the rate-control method for encoders. CBR is better suited for the fixed-bandwidth nature of networks, and it produces more predictable, stable video playback for client devices. With a consistent bitrate, it is easy for viewers to select a quality level that their connection can handle over time.

Depending on the complexity of the scene, VBR can result in spikes in bitrate, which can cause frame drops before the video reaches Amazon IVS and/or buffering in client players.

*We strongly recommend you only use CBR.* If you use VBR, your streams will be more subject to buffering and playback that is not smooth.

Use Progressive Signals

*Use progressive signal flows; avoid any interlaced video in production flow and/or encoding.* Progressive stream signals yield much better playback quality displaying a whole frame at a time, avoiding any motion artifacting that is produced when displaying an interlaced signal.
Network Requirements

You must have a stable internet connection that can maintain an adequate, constant upload stream. An unstable internet connection could result in stream stuttering and lagging for your viewers.

**Use wired connections.** WiFi and LTE connections can be spotty or suffer from interference or latency due to bad QoS/packet-queue prioritization. Whenever possible, rely on a hardwired connection for streams.

Plan to allocate 50% more bandwidth than the minimum required. The overhead is added to compensate for the bitrate fluctuations in encoding of a video bitstream.

Use a dedicated Internet VLAN to encoding machines. Keeping the encoder on a separate network prevents potentially disruptive effects, including: pollution by traffic, bandwidth bottlenecks and adverse security factors.

Closed Captioning

IVS supports closed captioning. As a streamer, if you want to offer captions to your audience, you must transmit caption data in an accepted format, either embedded in your stream or alongside your stream, through your video encoder.

Amazon IVS accepts captions in line 21 CEA-708/EIA-608 format (also referred to as 608 over 708). You can transmit captions using one of the following methods:

- CEA-708/EIA-608 embedded in the video elementary stream, as described in ATSC A/72 (SEI user_data). This format is common among television broadcast encoders.
- CEA-708/EIA-608 transmitted via RTMPS onCaptionInfo script/AMF0 tag. This format is common among Internet broadcast encoders and media servers like Elemental Technologies and Wowza. The Amazon IVS Player SDKs support one language; they do not support multi-track captions playback.

Note: The Amazon IVS Player SDKs support caption data only in the CC1 NTSC field 1. They do not support multi-track captions playback.

When transmitting via RTMPS, the payload must contain an ECMA array with two element pairs:

- A string named `type` that contains the characters `708`.
- A string named `data` that contains a base64-encoded CEA-708/EIA-608 payload.

For example:

```
00000000 12 00 00 69 00 00 00 00  
00000010 43 61 70 74 69 6f 02 00  
00000020 6e 6f 00 00 02 00 6d 6f  
6e 6f 00 00 02 00 6d 6f 6e 6f 00 00  
00000030 06 88 00 01 00 00 01 00  
00000040 00 00 00 00 00 00 00 00  
00000050 00 02 00 00 02 00 00 00  
00000060 00 00 00 00 00 00 00 00  
00000070 3d 00 00 00 00 00 00 74  
|...i...........on|
|CaptionInfo.....|
|..type...708..da|
|ta..<tQAxR0ESNAN|
|LAPyUrvyU1PyRQ7Pz|
|15fzes7PzVLPw0TPz|
|36fz04/xogPyUL/8|
|......t|
```

If you use the Elemental video encoder, set it up as follows:

- Set caption embed to “capture 608 Field 1.”
Stream from Windows Desktop with FFmpeg

FFmpeg is a free, open-source project comprising a vast suite of software libraries for handling video, audio, and other multimedia files and streams.

See the FFmpeg website for installation and other information about FFmpeg. Use the latest static build (do not compile).

After installing, choose an audio/video input source for FFmpeg. You can look up what is available, as follows:

```
ffmpeg -list_devices true -f dshow -i dummy
```

For more information, see here. Depending on what is available and what capture method is targeted, you should be able to capture the video/audio (embedded) directly from your selected device and encode the signals with FFmpeg. For example:

- **Webcam** — To capture output from the Logitech C920 webcam:

  ```
  ```

- **Video file** — FFmpeg works with many video-file formats and capture cards. Here is an example of streaming based on a MP4 input:

  ```
  ```

For more information about what to enter for `<IVS-ingest-server>` and `<IVS-stream-key>`, see the information about setting up live-streaming software in Getting Started with Amazon IVS (p. 3). For example:

- **Ingest server**: rtmps://jds34ksd3las.global-contribute.live-video.net/app/
- **Stream key**: sk_us-west-2_abcd1234efgh5678ijkl

Stream from Android and iOS with Larix Broadcaster

Larix Broadcaster is a free Android and iOS streaming client. It encodes video/audio data from cameras and microphones into H.264/AAC format, encapsulates the stream in FLV (Flash Video), and transmits it over RTMPS.

Larix is a pre-packaged application that is available for download from Google Play and the Apple App Store.

After downloading and installing it, configure Larix as follows:
1. Open Larix Broadcaster.
2. Agree to three system prompts asking you to **Allow Larix Broadcaster** to:
   a. Take pictures and record video
   b. Record audio
   c. Access photos, media, and files on your device
3. Under **Settings > Connections**, create a **New Connection**:
   a. Enter a configuration **Name** (anything you like).
   b. For **URL**, enter your Amazon IVS channel ingest server and stream key as one string:

   \[ \text{rtmps://<IVS-ingest-server>/app/<IVS-stream-key>>} \]

   See the information about setting up live-streaming software in **Getting Started with Amazon IVS** (p. 3).
4. Under **Settings > Video**:
   a. Set **Video Size** to 1080p resolution or less. We recommend 720p (1280x720).
   b. Set **FPS** to **30 fixed rate**.
   c. Under **Encoder**, set **Bitrate** to an appropriate bitrate to match your specified resolution (**Video Size**). For guidance, see **Resolution/Bitrate/FPS** (p. 116).
   d. Do not change the keyframe frequency; leave it set to 2 seconds, the default.
5. Exit **Settings** and return to the main view of the application. You should be able to start streaming to your channel on Amazon IVS.

**Testing the Stream**

**Always verify that your stream works.**

Navigate to the video stream in the **Amazon IVS console**, to watch what is being streamed and manage the live stream.
Amazon IVS Costs

There are separate costs for Amazon IVS live video and Amazon S3 storage related to the auto-record-to-S3 feature.

Live Video

The Amazon IVS pricing model incorporates separate fees for video input and output.

Video-input fees depend on your channel type (standard or basic):

- With a standard channel, multiple qualities are generated from the original input, to automatically give viewers the best experience for their devices and network conditions. Resolution can be up to 1080p and bitrate can be up to 8.5 Mbps. This is the default when you create a channel.
- With a basic channel, Amazon IVS delivers the original input to viewers. The viewer's video-quality choice is limited to the original input. Resolution can be up to 480p and bitrate can be up to 1.5 Mbps.

For video output, you pay an hourly rate for video delivered to viewers. Rates vary by resolution and "billing region" (where the video is delivered from). There are several tiers of video-output costs based on usage, including a free tier.

A useful interactive tool is the IVS Cost Estimator. You can plug in values for channel type, resolution, hours streamed, number of viewers, and billing region. When estimating costs, note the following rules of thumb:

- Viewers come and go, and on average, 50% of a stream is "delivered." The Cost Estimator includes a selector for "Average viewer watch duration." This defaults to 50%. Expect viewership for paid events to be higher; even in this case, though, it's likely that not all ticket-holders will view at the same time.
- Some viewers watch at a lower resolution than the source resolution of the broadcast. This is especially true for high-resolution streams: some viewers will watch at lower resolutions, which are less expensive. This is due to various viewer constraints, including bandwidth, network conditions, ISP, and hardware.
- Timing matters. For instance, if your stream competes with school, work, or vacation, this can affect your audience size.
- It is very hard to build a live audience from non-live users. Of course, there are exceptions; bringing in external talent (like influencers with their own following) can increase audience size.

Auto-Record to Amazon S3

There are no Amazon IVS charges for using the auto-record to Amazon S3 feature or for writing to S3. There are charges for Amazon S3 storage and for serving the stored video to viewers.

Storing Recorded Video

Customers can generate estimates of S3 storage needs and costs by using the IVS console. When a customer uses the console to set up recording for a channel (either when the channel is created or later),
a data-use estimator is offered. These data-use estimates can be plugged into the AWS Pricing Calculator for S3 to estimate the monthly cost of S3 storage and data movement.

In the console, when creating a new channel or editing an existing channel, select Auto-record to S3 in the Record and store streams area. This displays information about Associated costs.

Select Estimate data use to display the data-use calculator:
As noted on the screen, the estimates that are provided can be used with the AWS Pricing Calculator to compute estimates of the monthly cost incurred by S3 storage and data movement.

**Serving Recorded Video**

The cost of serving recorded video to viewers depends on the CDN that is used. For example, see the Amazon CloudFront pricing page.
# Glossary

| Term                          | Description                                                                                                                                                                                                                                                                                                                                 |
|-------------------------------|                                                                                                                                                                                                                                                                                                                                            |
| **ARN**                       | Amazon Resource Name. The following are general formats for ARNs; specific formats depend on the resource. To use an ARN, replace the italicized text (which is everything but `arn`) with the resource-specific information. Be aware that the ARNs for some resources omit the region, account ID, or both region and account ID.  
  
  `arn:partition:service:region:account-id:resource-id`  
  `arn:partition:service:region:account-id:resource-type/resource-id`  
  `arn:partition:service:region:account-id:resource-type:resource-id`                                                                                                                                                                                                 |
<p>| <strong>Broadcast, broadcaster</strong>    | Other terms for stream, streamer.                                                                                                                                                                                                                                                                                                         |
| <strong>Channel</strong>                   | An AWS resource that stores configuration for streaming. Streamers use the stream key associated with a channel to start a broadcast. All metrics and events are associated with a channel resource.                                                                                                                                                                                                 |
| <strong>Ingest</strong>                    | A streamer contributes or ingests a stream to Amazon IVS servers, using the RTMPS protocol, with a stream key in the URL.                                                                                                                                                                                                                 |
| <strong>Ingest server</strong>             | Identifies a specific Amazon IVS component that receives a stream, along with an ingestion protocol (RTMPS) and a path. Example: <code>rtmps://jds34ksdg3las.global-contribute.live-video.net/app/</code>                                                                                                                                               |
| <strong>JSON</strong>                      | JavaScript Object Notation. An open-standard file format that uses human-readable text to transmit data objects consisting of attribute-value pairs and array data types (or any other serializable value).                                                                                                                                                  |
| <strong>Latency, glass-to-glass latency</strong> | Latency is a delay in data transfer. Glass-to-glass latency refers to the delay from when a camera captures a live stream to when the stream appears on a viewer’s screen.                                                                                                                                |
| <strong>Playback key pair, playback token</strong> | Video playback may be restricted using playback-authorization tokens, which use public-key encryption. A playback key pair is the public-private pair of keys used to sign and validate the viewer authorization token for playback. For more information, see the Playback Key Pair endpoints in the Amazon IVS API Reference. |
| <strong>Playback URL</strong>              | Identifies the address a viewer uses to start playback for a specific channel. This address can be used globally. Amazon IVS automatically selects the best location on the Amazon IVS global content delivery network for each viewer, for delivering the video.                                                                 |
| <strong>RTMPS</strong>                     | The secure version of RTMP (Real-Time Messaging Protocol), running over TLS. RTMP is an industry standard for transmitting video over a network.                                                                                                                                                                                                 |</p>
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream key</td>
<td>An identifier assigned by Amazon IVS when you create a channel and used to authorize streaming. <em>Treat the stream key like a secret, since it allows anyone to stream to the channel.</em></td>
</tr>
<tr>
<td>Tag</td>
<td>A metadata label that you assign to an AWS resource. Tags can help you identify and organize your AWS resources. For more information, see “Tagging” in the Amazon IVS API Reference.</td>
</tr>
<tr>
<td>Timed metadata</td>
<td>Metadata with timestamps. It can be inserted into a stream programmatically, using the Amazon IVS API. When Amazon IVS processes a stream, the timed metadata is synchronized with the audio and video frames. During playback, all viewers of the stream get the metadata. The timecode serves as a cue point, which can be used to trigger an action based on the data; for example, updating player statistics on the page (for a sports stream).</td>
</tr>
<tr>
<td>Transmux</td>
<td>A simple repackaging of an ingested stream to Amazon IVS, with no re-encoding of the video stream. &quot;Transmux&quot; is short for transcode multiplexing, a process that changes the format of an audio and/or video file while keeping some or all of the original streams. Transmuxing converts to a different container format without changing the file contents. Distinguished from a transcode.</td>
</tr>
<tr>
<td>Transcode</td>
<td>Transcoding is a process that converts video and audio from one format to another. An incoming stream may be transcoded to a different format at multiple bitrates and resolutions, to support a range of playback devices and network conditions.</td>
</tr>
<tr>
<td>View</td>
<td>A unique viewing session which is actively downloading or playing video. A view starts when a viewing session begins video playback. A view ends when a viewing session stops video playback. Playback is the sole indicator of viewership; engagement heuristics such as audio levels, browser tab focus, and video quality are not considered. When counting views, Amazon IVS does not consider the legitimacy of individual viewers or try to deduplicate localized viewership, such as multiple video players on a single machine.</td>
</tr>
</tbody>
</table>
## Document History

### User Guide Changes

<table>
<thead>
<tr>
<th>Update History Change</th>
<th>Update History Description</th>
<th>Update History Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streaming Configuration (p. 115)</td>
<td>In &quot;Audio Settings,&quot; specified a minimum bitrate, 96 Kbps.</td>
<td>September 22, 2021</td>
</tr>
<tr>
<td>Getting Started with Amazon IVS (p. 16)</td>
<td>In &quot;Step 4: Set Up Streaming Software,&quot; added a note about disconnecting if no data is sent for 30 seconds.</td>
<td>September 20, 2021</td>
</tr>
<tr>
<td>Identity-based policy example (p. 126)</td>
<td>In Amazon IVS Security, fixed a typo in the example in Access an Amazon IVS Channel: added ending punctuation (} }}).</td>
<td>September 17, 2021</td>
</tr>
<tr>
<td>SDK sizes for Player 1.4.1 and 1.4.0 (p. 126)</td>
<td>In the Release Notes for Player 1.4.1 and 1.4.0, we made corrections to the tables of mobile SDK sizes.</td>
<td>September 16, 2021</td>
</tr>
<tr>
<td>Player 1.4.1 release (p. 126)</td>
<td>Bug-fix release; see Amazon IVS Release Notes. Also updated version number and artifact links in all player guides: Web, Android, iOS, Video.js Integration, and JW Player Integration.</td>
<td>September 8, 2021</td>
</tr>
<tr>
<td></td>
<td>On the Amazon IVS documentation landing page, updated the player SDK Reference links to point to the new versions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Streaming Configuration, updated the information on Closed Captioning.</td>
<td></td>
</tr>
<tr>
<td>Broadcasting: SDK for Android Guide (p. 22)</td>
<td>In &quot;Set the ImagePreviewView for Preview,&quot; made minor text clarifications. In &quot;Swap Cameras,&quot; fixed two typos. In &quot;Create a Broadcast Configuration,&quot; deleted the line referencing video.setDefaultAspectRatio, which is not usable now.</td>
<td>September 1, 2021</td>
</tr>
<tr>
<td>Streaming configuration with FFmpeg (p. 119)</td>
<td>Modified settings for capturing video files. Specifically, –g 120 was changed to</td>
<td>August 23, 2021</td>
</tr>
</tbody>
</table>
-force_key_frames
expr:gte(t,n_forced*2).
This causes the encoder to insert
a keyframe every 2 seconds,
regardless of source-input frame
rate.

<table>
<thead>
<tr>
<th>Amazon IVS Player: SDK for Web Guide (p. 44)</th>
<th>Added new &quot;Known Issue&quot; for Pixel 4/4a mobile browsers.</th>
<th>August 20, 2021</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon IVS Player: Video.js Integration (p. 53)</td>
<td>In &quot;Sample Code,&quot; updated the version number to 7.14.3. There is a security vulnerability in versions of Video.js earlier than 7.14.3.</td>
<td>August 19, 2021</td>
</tr>
<tr>
<td>Streaming Configuration (p. 117)</td>
<td>For the STANDARD channel type, added a note that audio is transcoded only for renditions 360p and below; above that, audio is passed through.</td>
<td>August 18, 2021</td>
</tr>
<tr>
<td>Getting Started with Amazon IVS (p. 3)</td>
<td>In &quot;Step 2: Set Up IAM Permissions,&quot; added steps to attach the policy to an existing user. This new procedure is in addition to the old procedure, which is for creating a new user and attaching a policy to that user.</td>
<td>August 11, 2021</td>
</tr>
<tr>
<td>Player 1.4.0 release (p. 126)</td>
<td>Updated version number and artifact links for the new release, in all player guides: Web, Android, iOS, Video.js Integration, and JW Player Integration. On the Amazon IVS documentation landing page, updated the player SDK Reference links to point to the new versions. Also see the Amazon IVS Release Notes (p. 142) for this release.</td>
<td>August 10, 2021</td>
</tr>
<tr>
<td>Amazon IVS Player: SDK for Web Guide (p. 41)</td>
<td>In &quot;Setup With NPM,&quot; added a note about hosting player static assets from your own domain.</td>
<td>July 30, 2021</td>
</tr>
<tr>
<td>Topic</td>
<td>Changes</td>
<td>Date</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Getting Started with Amazon IVS (p. 3)</td>
<td>In “Step 2: Set Up IAM Permissions,” updated policy information and instructions.</td>
<td>July 29, 2021</td>
</tr>
<tr>
<td></td>
<td>In “Step 3: Create a Channel with Optional Recording,” added a section, “Auto-Record to S3” (to replace a prior paragraph).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In “Step 4: Set up Streaming Software,” added a section, “Streaming with the Amazon IVS Broadcast SDK.”</td>
<td></td>
</tr>
<tr>
<td>Auto-Record to S3 (p. 77)</td>
<td>Added a new section, “Playback of Recorded Content from Private Buckets.” Also updated the introduction to this page.</td>
<td>July 28, 2021</td>
</tr>
<tr>
<td>Amazon IVS Broadcast SDK (Android and iOS) (p. 126)</td>
<td>Initial release of the broadcast SDK for Android and iOS. See the documentation under “Amazon IVS Broadcast SDK,” a new section of the Amazon IVS documentation landing page.</td>
<td>July 27, 2021</td>
</tr>
<tr>
<td>Amazon IVS Player (p. 126)</td>
<td>Updated Desktop Browsers to indicate Amazon IVS Player 1.3.0 support for ultra-low latency on new versions of Safari for macOS.</td>
<td>July 14, 2021</td>
</tr>
<tr>
<td>Amazon IVS Service Quotas (p. 111)</td>
<td>For the PutMetadata endpoint, added a limit of 155 TPS per account.</td>
<td>June 29, 2021</td>
</tr>
<tr>
<td>ivs.rocks (p. 126)</td>
<td>On the Amazon IVS User Guide landing page, added a link to and brief description of ivs.rocks.</td>
<td>June 25, 2021</td>
</tr>
<tr>
<td>Player Browser &amp; Platform Requirements (p. 39)</td>
<td>For the Amazon IVS Player, added links to sites listing the latest versions of supported browsers.</td>
<td>June 25, 2021</td>
</tr>
<tr>
<td>Streaming Configuration (p. 115)</td>
<td>In “Channel Types,” updated definitions of channel types. For STANDARD channels, resolution can be up to 1080p; for BASIC channels, 480p. (The prior definitions were only in terms of vertical resolution.)</td>
<td>June 17, 2021</td>
</tr>
<tr>
<td>Costs (p. 121)</td>
<td>Added a new page on costs.</td>
<td>June 17, 2021</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Player mobile-browser support (p. 126)</td>
<td>In Mobile Browsers, added information about support for Chrome for iPadOS and Safari for iPadOS.</td>
<td>June 14, 2021</td>
</tr>
<tr>
<td>Player SDK size (p. 126)</td>
<td>Added a new &quot;SDK Size&quot; section to the Android and iOS Player SDK guides.</td>
<td>June 11, 2021</td>
</tr>
<tr>
<td>Amazon IVS Player: SDK for Web Guide (p. 44)</td>
<td>Added two &quot;Known Issues&quot; when playing content on an iOS mobile browser (with player.getQualities() and player.getLiveLatency() calls).</td>
<td>June 9, 2021</td>
</tr>
<tr>
<td>Supported regions and service endpoints (p. 126)</td>
<td>Replace lists of supported regions with a link to the Amazon IVS page in the AWS General Reference, which is updated automatically when support for new regions is added. Changes were made on the Monitoring Amazon IVS with Amazon CloudWatch page.</td>
<td>June 8, 2021</td>
</tr>
<tr>
<td>Amazon IVS Player issues (p. 126)</td>
<td>In &quot;Known Issues and Workarounds,&quot; for the Web, Android, and iOS Player, asked customers to report all issues to Support. Also added an issue with Android 11 emulators.</td>
<td>June 4, 2021</td>
</tr>
<tr>
<td>Android and iOS Player 1.3.3 release (p. 126)</td>
<td>Bug-fix release; see Amazon IVS Release Notes. Also updated version-number references in links and text in the Android Player Guide and iOS Player Guide.</td>
<td>June 1, 2021</td>
</tr>
<tr>
<td>Setting Up Private Channels (p. 73)</td>
<td>Updated “Generate and Sign Playback Tokens” (information on creating the signature and steps in &quot;Instructions&quot;).</td>
<td>May 26, 2021</td>
</tr>
<tr>
<td>Global versus regional (p. 126)</td>
<td>Moved &quot;Global Solution, Regional Control&quot; from Getting Started with Amazon IVS to What Is Amazon IVS.</td>
<td>May 21, 2021</td>
</tr>
<tr>
<td>Amazon IVS Player: Video.js Integration (p. 53)</td>
<td>In &quot;Sample Code,&quot; updated the Cloudflare version number from 7.6.6 to 7.11.4.</td>
<td>May 20, 2021</td>
</tr>
</tbody>
</table>

The Amazon IVS documentation landing page always points to the most current versions of the Player SDK References.
<table>
<thead>
<tr>
<th><strong>Amazon Interactive Video Service User Guide</strong></th>
<th><strong>User Guide Changes</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Android Player 1.3.2 release (p. 126)</strong></td>
<td>Bug-fix release; see Amazon IVS Release Notes. Also updated version-number references in links and text in the Android Player Guide.</td>
</tr>
<tr>
<td><strong>Amazon IVS Service Quotas (p. 111)</strong></td>
<td>Minor wording changes. Deleted information about maximum number of tags; this was moved to the API Reference.</td>
</tr>
<tr>
<td><strong>Amazon IVS Release Notes (p. 142)</strong></td>
<td>Added a note for Web Player 1.3.1: the 1.3.0 NPM package exists but does not work.</td>
</tr>
<tr>
<td><strong>Using Amazon EventBridge with Amazon IVS (p. 90)</strong></td>
<td>Updated stream_id to be a &quot;sanitized&quot; value in all relevant examples.</td>
</tr>
<tr>
<td><strong>Amazon IVS Player: SDK for Web Guide (p. 44)</strong></td>
<td>Added a known issue and workaround, for player.seekTo() calls when playing recorded content on an iOS mobile browser.</td>
</tr>
<tr>
<td><strong>Streaming Configuration (p. 115)</strong></td>
<td>Renamed the Encoder Configuration page to Streaming Configuration.</td>
</tr>
<tr>
<td><strong>Using Amazon EventBridge with Amazon IVS (p. 90)</strong></td>
<td>In &quot;Examples: Recording State Change,&quot; added the recording_duration_ms field, changed the example value of the recording_s3_key_prefix field, and changed the value of the recording_status_reason field.</td>
</tr>
<tr>
<td><strong>User Guide Changes</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| **Player 1.3 release (p. 126)** | Updated version number and artifact links for the new release, in all player guides: Web, Android, iOS, Video.js Integration, and JW Player Integration. For Android, added mavenCentral() to "Install the Library."  
On the Amazon IVS documentation landing page, updated the player SDK Reference links to point to the new versions.  
In Player 1.3.0 and later, timed metadata is now supported on Chrome and Safari for iOS. This is noted in the Amazon IVS Player (p. 38) overview (table on "Mobile Browsers") and Embedding Metadata within a Video Stream (p. 65) (in "Consuming Metadata").  
Also see the Amazon IVS Release Notes (p. 142) for this release. | May 5, 2021 |
| **Amazon IVS Service Quotas (p. 113)** | Added a new section, "Service Quotas Integration with CloudWatch Usage Metrics." | April 26, 2021 |
| **Maximum duration of a stream (p. 126)** | In Getting Started with Amazon IVS ("Step 4: Set Up Streaming Software"), added a note about the maximum duration of a stream, 48 hours. | April 23, 2021 |
| **IAM policy changes (p. 126)** | Made several IAM policy changes:  
- **Getting Started with Amazon IVS (p. 3)** — In "Step 2: Set Up IAM Permissions," added service quotas.  
- **Amazon IVS Security** — In "Use the Amazon IVS Console," simplified the policy example and added service quotas. | April 22, 2021 |
<table>
<thead>
<tr>
<th>User Guide Changes</th>
<th>Various doc changes for the release of new CloudWatch metrics:</th>
</tr>
</thead>
<tbody>
<tr>
<td>New CloudWatch metrics (p. 126)</td>
<td>- Monitoring Amazon IVS with Amazon CloudWatch (p. 93)</td>
</tr>
<tr>
<td></td>
<td>- Added new metrics: concurrent views and concurrent streams.</td>
</tr>
<tr>
<td></td>
<td>- Service Quotas (p. 111) - Updated the names of related quotas to match the new metrics.</td>
</tr>
<tr>
<td></td>
<td>- Glossary (p. 124) - Added &quot;view.&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Guide Changes</th>
<th>New User Guide page for this new Amazon IVS functionality. This also affects several existing documents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-Record to S3 (p. 77)</td>
<td>- Getting Started with Amazon IVS - Added IAM policy info for R2S3. Rewrote the step to create a channel. Added a paragraph on optionally enabling local recording in OBS Studio. New section on disabling recording.</td>
</tr>
<tr>
<td></td>
<td>- Using Amazon EventBridge with Amazon IVS - Added Recording State Change events.</td>
</tr>
<tr>
<td></td>
<td>- Monitoring Amazon IVS with Amazon CloudWatch - Added RecordedTime metric.</td>
</tr>
<tr>
<td></td>
<td>- Amazon IVS Security - Added a section on &quot;Using Service-Linked Roles (SLRs) for Amazon IVS.&quot;</td>
</tr>
<tr>
<td></td>
<td>- Service Quotas - Added &quot;API Call Rate Quotas&quot; for the new recording-configuration endpoints and a &quot;Resource Quotas&quot; limit for recording configurations.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Guide Changes</th>
<th>In &quot;Closed Captioning,&quot; clarified that the Player SDKs support only 1 language, not multi-track captions playback.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon IVS Streaming Configuration (p. 118)</td>
<td></td>
</tr>
</tbody>
</table>

April 13, 2021
April 7, 2021
March 29, 2021
<table>
<thead>
<tr>
<th>Change Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global versus regional (p. 126)</td>
<td>March 25, 2021</td>
</tr>
<tr>
<td>In What is Amazon IVS, added a new section, &quot;Global Solution, Regional Control,&quot; to clarify what is global versus regional. In Getting Started with Amazon IVS, mentioned selecting a region, in the instructions for creating a channel.</td>
<td></td>
</tr>
<tr>
<td>EventBridge event latency &amp; IDR/Keyframe encoder setting (p. 126)</td>
<td>March 25, 2021</td>
</tr>
<tr>
<td>Clarified the relationship between the IDR/Keyframe video-encoder setting and latency in some EventBridge events. This affects two documents:</td>
<td></td>
</tr>
<tr>
<td>• &quot;Amazon IVS Streaming Configuration&quot; (p. 115) — See the IDR/Keyframe bullet in &quot;Reducing Latency.&quot;</td>
<td></td>
</tr>
<tr>
<td>• &quot;Using Amazon EventBridge with Amazon IVS&quot; (p. 85) — See the new &quot;Note on latency of Stream State Change events.&quot;</td>
<td></td>
</tr>
<tr>
<td>Monitoring Amazon IVS with Amazon CloudWatch (p. 93)</td>
<td>March 18, 2021</td>
</tr>
<tr>
<td>Clarified how long CloudWatch retains data.</td>
<td></td>
</tr>
<tr>
<td>Streaming Configuration (p. 117)</td>
<td>March 15, 2021</td>
</tr>
<tr>
<td>In &quot;Audio Settings,&quot; changed the supported bitrate to 320 Kpbs (from 192).</td>
<td></td>
</tr>
<tr>
<td>Required versions of TLS (p. 126)</td>
<td>March 15, 2021</td>
</tr>
<tr>
<td>Clarified requirements for TLS (Transport Layer Security). For API calls, clients must support TLS 1.0 or later, but we recommend TLS 1.2 or later. For streaming/playback, TLS version 1.2 or later is required.</td>
<td></td>
</tr>
<tr>
<td>Changes were made in two documents: Streaming Configuration (section on &quot;Stream Ingest: Codecs, RTMPS, and Port 443&quot;) and Security (section on &quot;Infrastructure Security&quot;).</td>
<td></td>
</tr>
<tr>
<td>Amazon IVS Player: SDK for Web Guide (p. 44)</td>
<td>March 15, 2021</td>
</tr>
<tr>
<td>Added a known issue with HTML5 and setQuality().</td>
<td></td>
</tr>
<tr>
<td>Amazon IVS Player: SDK for Web Guide (p. 44)</td>
<td>March 11, 2021</td>
</tr>
<tr>
<td>Added a known issue with captions.</td>
<td></td>
</tr>
<tr>
<td>User Guide Changes</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Amazon IVS Player (p. 126)</strong></td>
<td></td>
</tr>
<tr>
<td>Also, for Android, noted that after the <code>player.release()</code> method is called, the player can no longer be used.</td>
<td></td>
</tr>
<tr>
<td><strong>Monitoring Amazon IVS with Amazon CloudWatch (p. 93)</strong></td>
<td></td>
</tr>
<tr>
<td>Updated the procedure for accessing Amazon IVS metrics using the CloudWatch console: added information on when “IVS” is listed and a screenshot.</td>
<td></td>
</tr>
<tr>
<td><strong>Security (p. 110)</strong></td>
<td></td>
</tr>
<tr>
<td>In “Infrastructure Security,” added a note that Amazon IVS streaming requires TLS 1.2. Also listed a new web page for details on AWS global network security procedures.</td>
<td></td>
</tr>
<tr>
<td><strong>Amazon IVS Player: JW Player Integration (p. 62)</strong></td>
<td></td>
</tr>
<tr>
<td>New User Guide page on the JW Player plug-in for the Amazon IVS player. Also added a JW Player row to the Framework Integrations table in the Web Player Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Using Amazon EventBridge with Amazon IVS (p. 85)</strong></td>
<td></td>
</tr>
<tr>
<td>Expanded the wording about guarantees for sending events.</td>
<td></td>
</tr>
<tr>
<td><strong>Using Amazon EventBridge with Amazon IVS (p. 85)</strong></td>
<td></td>
</tr>
<tr>
<td>Added: Events are sent on a best-effort basis.</td>
<td></td>
</tr>
<tr>
<td><strong>Streaming Configuration (p. 117)</strong></td>
<td></td>
</tr>
<tr>
<td>Changed the codec audio setting from AAC to AAC (LC).</td>
<td></td>
</tr>
<tr>
<td><strong>Amazon IVS Service Quotas (p. 112)</strong></td>
<td></td>
</tr>
<tr>
<td>In &quot;Resource Quotas,&quot; added the maximum number of tags for a resource.</td>
<td></td>
</tr>
<tr>
<td><strong>Android Player 1.2.1 release (p. 126)</strong></td>
<td></td>
</tr>
<tr>
<td>Bug-fix release; see Amazon IVS Release Notes. Also updated version-number references in links and text in the Android Player Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Amazon IVS Release Notes (p. 142)</strong></td>
<td></td>
</tr>
<tr>
<td>For Amazon IVS Android Player 1.2.0 and 1.1.0, added a known issue which causes the SDK to crash.</td>
<td></td>
</tr>
<tr>
<td><strong>Getting Started with Amazon IVS (p. 6)</strong></td>
<td></td>
</tr>
<tr>
<td>In bullet on playback URLs (in &quot;Step 3: Create a Channel&quot;), added a note that custom domains for playback are not supported.</td>
<td></td>
</tr>
<tr>
<td><strong>Amazon IVS Release Notes (p. 142)</strong></td>
<td>Deleted download links for iOS Player 1.0.6 and 1.0.0; these versions are deprecated.</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Added a &quot;Known Issue&quot; for iOS Player 1.2.0.</td>
</tr>
<tr>
<td><strong>Player 1.2.0 release (p. 126)</strong></td>
<td>Updated version number and artifact links for the new release, in all player guides: Web, Android, iOS, and Video.js Integration. Added a Known Issue to the Android guide.</td>
</tr>
<tr>
<td></td>
<td>On the Amazon IVS documentation landing page, updated the player SDK Reference links to point to the new versions.</td>
</tr>
<tr>
<td></td>
<td>Also see the updated Amazon IVS Release Notes.</td>
</tr>
<tr>
<td><strong>Setting Up Private Channels (p. 73)</strong></td>
<td>In the section on &quot;Generate and Sign Playback Tokens,&quot; the channel-arn value in the JWT payload is a string.</td>
</tr>
<tr>
<td><strong>Using Amazon EventBridge with Amazon IVS (p. 85)</strong></td>
<td>Added stream_id field to many events. This is a unique stream identifier assigned each time a channel goes live. For a given channel, each live stream has a new stream_id. Stream IDs allow customers to distinguish different stream sessions on the same channel.</td>
</tr>
<tr>
<td><strong>Embedding Metadata Within a Video Stream (p. 68)</strong></td>
<td>Added new section on &quot;Viewing Timed Metadata&quot; from the Amazon IVS console.</td>
</tr>
<tr>
<td><strong>Web Player Guide (p. 43)</strong></td>
<td>Updated the section on &quot;Content Security Policy,&quot; especially for hosting assets on a separate page when using Safari.</td>
</tr>
<tr>
<td><strong>Service Quotas (CCV and CCB limits) (p. 126)</strong></td>
<td>Added notes about the importance of ensuring adequate concurrent-viewer and concurrent-broadcaster limits, especially before large streaming events. See Getting Started with Amazon IVS and Amazon IVS Service Quotas.</td>
</tr>
<tr>
<td>User Guide Changes</td>
<td>Date</td>
</tr>
<tr>
<td>-------------------</td>
<td>------</td>
</tr>
<tr>
<td><strong>Using Amazon EventBridge with Amazon IVS (p. 88)</strong></td>
<td>October 28, 2020</td>
</tr>
<tr>
<td>Updated Limit Breach events: the detail section of the JSON blob uses <code>limit_name</code> for all these events. (Previously only Concurrent Broadcasts showed that and the others showed <code>limit</code>.)</td>
<td></td>
</tr>
<tr>
<td><strong>Setting Up Private Channels (p. 73)</strong></td>
<td>October 27, 2020</td>
</tr>
<tr>
<td>In the section on &quot;Generate and Sign Playback Tokens,&quot; noted that the <code>exp</code> (expiration) field in JWT payloads is an integer.</td>
<td></td>
</tr>
<tr>
<td><strong>Amazon IVS Service Quotas (p. 111)</strong></td>
<td>October 27, 2020</td>
</tr>
<tr>
<td>Increased three limits: number of channels, concurrent viewers, and concurrent broadcasts.</td>
<td></td>
</tr>
<tr>
<td><strong>Web Player 1.1.2 release (p. 126)</strong></td>
<td>October 9, 2020</td>
</tr>
<tr>
<td>Bug-fix release; see the Amazon IVS Release Notes. Version-number references in links and text were updated in the Web Player Guide and Video.js integration Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Ingest resolution quotas &amp; event (p. 126)</strong></td>
<td>October 9, 2020</td>
</tr>
<tr>
<td>Added service quotas and EventBridge events for ingest resolution. See Amazon IVS Service Quotas and Using Amazon EventBridge with Amazon IVS.</td>
<td></td>
</tr>
<tr>
<td><strong>Player 1.1.0 release (p. 126)</strong></td>
<td>October 7, 2020</td>
</tr>
<tr>
<td>Updated version number and artifact links for the new release, in all player guides: Web, Android, iOS, and Video.js Integration. In the iOS and Web guides, added a new section on &quot;Known Issues.&quot; On the Amazon IVS documentation landing page, updated the player SDK Reference links to point to the new versions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### User Guide Changes

<table>
<thead>
<tr>
<th>Topic</th>
<th>Changes</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting Up Private Channels (p. 69)</td>
<td>Added a new section, &quot;Workflow for Private Channels.&quot; In the section on generating and signing tokens, clarified payload field descriptions and example. Corrected examples for listing and getting playback key pairs.</td>
<td>September 21, 2020</td>
</tr>
<tr>
<td>Using Amazon EventBridge with Amazon IVS (p. 85)</td>
<td>The channel_name field was added to several events.</td>
<td>September 14, 2020</td>
</tr>
<tr>
<td>Embedding Metadata Within a Video Stream (p. 65)</td>
<td>Expanded information on setting up IAM permissions (full procedure and policy), inserting metadata (added a CLI procedure), and consuming metadata (linked to several GitHub demos).</td>
<td>September 14, 2020</td>
</tr>
<tr>
<td>Player guides (p. 126)</td>
<td>Clarified which is the most current version of each player (Web, Android, IOS, and Video.js Integration).</td>
<td>September 9, 2020</td>
</tr>
<tr>
<td>Getting Started with Amazon IVS (p. 18)</td>
<td>Mentioned that there is a short delay before a new stream can be viewed in the console.</td>
<td>September 9, 2020</td>
</tr>
<tr>
<td>Amazon IVS Release Notes (p. 156)</td>
<td>Changed the Player iOS download link to be the same as what is in the Player iOS Guide.</td>
<td>September 9, 2020</td>
</tr>
<tr>
<td>Embedding Metadata within a Video Stream (p. 65)</td>
<td>Added link to relevant AWS blog posts.</td>
<td>September 3, 2020</td>
</tr>
<tr>
<td>Amazon IVS Player (p. 38)</td>
<td>Expanded the discussion of player features. Clarified that we can guarantee the performance of only the Amazon IVS player (not third-party players).</td>
<td>September 3, 2020</td>
</tr>
<tr>
<td>Amazon IVS Service Quotas (p. 111)</td>
<td>Corrected this to indicate that only the channels, concurrent viewers, and concurrent broadcasts quotas can be adjusted.</td>
<td>August 31, 2020</td>
</tr>
<tr>
<td>Streaming Configuration (p. 115)</td>
<td>Several changes, including adding Reducing Latency subsection on &quot;Avoid Third-Party Streaming/Forwarding Services&quot; and clarifying why we strongly recommend CBR over VBR.</td>
<td>August 24, 2020</td>
</tr>
<tr>
<td>Embedding Metadata within a Video Stream (p. 65)</td>
<td>Updated Web example in Consuming Timed Metadata.</td>
<td>August 24, 2020</td>
</tr>
<tr>
<td>Amazon IVS Player: SDK for Android Guide (p. 44)</td>
<td>Updated code example in Install the Library.</td>
<td>August 24, 2020</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Using Amazon EventBridge with Amazon IVS (p. 88)</td>
<td>In the section on “Examples: Limit Breach,” updated several field names: limit_name, limit_value, exceeded_by, and limit_unit. These names include underscores (not dashes).</td>
<td>August 19, 2020</td>
</tr>
<tr>
<td>Setting Up Private Channels (p. 69)</td>
<td>New User Guide page on new Amazon IVS functionality, supporting private channels. This also affects several existing documents:</td>
<td>August 19, 2020</td>
</tr>
<tr>
<td></td>
<td>Getting Started with Amazon IVS and Logging Amazon IVS API Calls with AWS CloudTrail: Added authorized field to channel.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Security: Several changes including a new section on “Privileged and Unprivileged Access.”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Service Quotas: Added several playback quotas.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Glossary: Added playback key pair.</td>
<td></td>
</tr>
<tr>
<td>Getting Started with Amazon IVS (p. 3)</td>
<td>Added a new section on AWS Regional Service.</td>
<td>August 11, 2020</td>
</tr>
<tr>
<td>Amazon IVS Player: SDK for iOS Guide (p. 48)</td>
<td>Updated links to the reference documentation and framework download to point to the 1.0.6 release. Also updated reference-doc link on the Amazon IVS doc landing page.</td>
<td>August 11, 2020</td>
</tr>
<tr>
<td>Using Amazon EventBridge with Amazon IVS (p. 86)</td>
<td>Amazon IVS EventBridge events are now available through the Amazon EventBridge console. See the section on &quot;Creating Amazon EventBridge Rules for Amazon IVS.&quot;</td>
<td>August 5, 2020</td>
</tr>
<tr>
<td>Amazon IVS Player: Video.js Integration (p. 54)</td>
<td>In the “Setup With NPM” section, updated the link to the Video.js npm package to install, to version 7.6.6.</td>
<td>July 30, 2020</td>
</tr>
</tbody>
</table>
Using Amazon EventBridge with Amazon IVS (p. 85)

For Amazon IVS stream-state and stream-health changes, the event name is provided in a field called `event_name` (not `eventName`, as previously documented).

July 29, 2020

Getting Started with Amazon IVS (p. 16)

Changed the instructions for setting up streaming software, to indicate that port 443 is required for Amazon IVS ingest. This also affects the Streaming Configuration document; see the new section on RTMPs and Port 443.

July 27, 2020

Amazon IVS Player: SDK for iOS Guide (p. 48)

Changed the download location of the latest version, in the instructions for installing the framework manually.

July 27, 2020

Embedding Metadata Within a Video Stream (p. 65)

Added Android and iOS examples of consuming timed metadata.

July 24, 2020

New service and User Guide (p. 1)

This is the initial release of Amazon Interactive Video Service (IVS).

July 15, 2020

API Reference Changes

<table>
<thead>
<tr>
<th>API Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format of time fields</td>
<td>Updated the description of <code>startTime</code> in the Stream and StreamSummary objects, to add that it's an ISO 8601 timestamp returned as a string.</td>
<td>September 21, 2021</td>
</tr>
<tr>
<td>STANDARD channel type</td>
<td>For the STANDARD channel type, added notes that audio is transcoded only for renditions 360p and below; above that, audio is passed through.</td>
<td>August 18, 2021</td>
</tr>
<tr>
<td>ListTagsForResource endpoint</td>
<td>Removed support for pagination; i.e., the <code>maxResults</code> request field and <code>nextToken</code> request/response field. (Pagination did not work correctly.)</td>
<td>August 13, 2021</td>
</tr>
<tr>
<td>PutMetadata TPS limit per account</td>
<td>For the PutMetadata endpoint, added a limit of 155 TPS per account.</td>
<td>June 29, 2021</td>
</tr>
<tr>
<td>Channel-type definitions</td>
<td>Updated the definitions of channel types. For STANDARD channels, resolution can be up to 1080p; for BASIC channels, 480p. (The prior definitions were only in terms of vertical resolution.)</td>
<td>June 17, 2021</td>
</tr>
<tr>
<td>API Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Supported regions and service</td>
<td>Replace lists of supported regions with a link to the Amazon IVS page in the AWS General Reference, which is updated automatically when support for new regions is added. Changes were made on the &quot;Welcome&quot; page.</td>
<td>June 8, 2021</td>
</tr>
<tr>
<td>endpoints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tagging</td>
<td>In &quot;Tagging&quot; (in the &quot;Welcome&quot; section), added the maximum number of tags that can be applied to a resource (50).</td>
<td>May 12, 2021</td>
</tr>
<tr>
<td>New CloudWatch Metrics</td>
<td>Changed the definition of viewerCount in the Stream and StreamSummary objects.</td>
<td>April 13, 2021</td>
</tr>
</tbody>
</table>
| Auto-Record to S3                | • Added 4 recording-configuration endpoints (Create, Delete, Get, List).  
• Add 4 data types (DestinationConfiguration, RecordingConfiguration, RecordingConfigurationSummary, S3DestinationConfiguration).  
• Added a RecordingConfigurationArn field to the Channel and ChannelSummary objects and channel endpoints.  
• Modified ListChannels to filter by recording-configuration ARN.                                                                                                                                              | April 7, 2021   |
| Authentication & authorization   | • Added an “Authentication versus Authorization” section to clarify the difference between these concepts.  
• Changed the description of the authorized field (in the Channel data type and channel endpoints), to: "Whether the channel is private (enabled for playback authorization)."                                                             | March 16, 2021  |
| PutMetadata                      | Added a minimum length (1) for the metadata request field.                                                                                                                                                     | March 4, 2021   |
| Channel latency mode             | In Create/UpdateChannel and Channel/ChannelSummary objects, added a description of latencyMode values.                                                                                                                                                                   | December 18, 2020|
| Channel default values           | • In Channel data type, add default value for authorized.  
• In Channel data type and CreateChannel, add default value for type.                                                                                                                                                                                              | December 17, 2020|
<p>| All List endpoints               | Indicated that the maxResults request field has a default value, 50.                                                                                                                                                                                                   | December 5, 2020|
| Stream &amp; StreamSummary objects   | Changed the description of the viewerCount field to indicate that a value of -1 indicates that the request timed out; in this case, retry.                                                                                                                                   | November 10, 2020|
| Authentication                    | Added Sigv4 signing info. See &quot;Authentication&quot; in the Welcome section.                                                                                                                                                                                                | October 9, 2020  |</p>
<table>
<thead>
<tr>
<th>API Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeleteChannel &amp; DeleteStreamKey</td>
<td>Changed the HTTP response from 200 to 204.</td>
<td>August 26, 2020</td>
</tr>
<tr>
<td>DeleteChannel</td>
<td>Clarified how to delete a channel that’s live, to avoid an error.</td>
<td>August 20, 2020</td>
</tr>
</tbody>
</table>
| Playback authorization (for private channels) | • New PlaybackKeyPair endpoints  
• A new authorized field in the Channel and ChannelSummary objects  
• New objects, PlaybackKeyPair and PlaybackKeyPairSummary | August 19, 2020 |
| New service and API Reference      | This is the initial release of Amazon Interactive Video Service (IVS).                           | July 15, 2020   |
Amazon Interactive Video Service Release Notes

September 8, 2021

Amazon IVS Player SDK 1.4.1

<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Fixed the closed-caption decoder to handle captions that are inserted out of order.</td>
</tr>
</tbody>
</table>
| Web player 1.4.1 (p. 41) & Video.js integration (p. 53) & JW player integration (p. 62) | NPM Package: https://www.npmjs.com/package/amazon-ivs-player  
Script asset: https://player.live-video.net/1.4.1/amazon-ivs-player.min.js  
Video.js tech asset: https://player.live-video.net/1.4.1/amazon-ivs-videojs-tech.min.js  
Reference documentation: https://aws.github.io/amazon-ivs-player-docs/1.4.1/web/ |
| Android player 1.4.1 (p. 44) | Reference documentation: https://aws.github.io/amazon-ivs-player-docs/1.4.1/android/ |
| iOS Player 1.4.1 (p. 48) | Download: https://player.live-video.net/1.4.1/AmazonIVSPlayer.xcframework.zip  
Reference documentation: https://aws.github.io/amazon-ivs-player-docs/1.4.1/ios/ |

Mobile SDK Size: Android

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Compressed Size</th>
<th>Uncompressed Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>arm64-v8a</td>
<td>1.00 MB</td>
<td>2.79 MB</td>
</tr>
<tr>
<td>armeabi-v7a</td>
<td>0.83 MB</td>
<td>2.15 MB</td>
</tr>
<tr>
<td>x86_64</td>
<td>1.11 MB</td>
<td>3.06 MB</td>
</tr>
<tr>
<td>x86</td>
<td>1.11 MB</td>
<td>2.94 MB</td>
</tr>
</tbody>
</table>
Mobile SDK Size: iOS

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Compressed Size</th>
<th>Uncompressed Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>universal</td>
<td>0.89 MB</td>
<td>1.91 MB</td>
</tr>
<tr>
<td>arm64</td>
<td>0.45 MB</td>
<td>1.05 MB</td>
</tr>
<tr>
<td>armv7</td>
<td>0.44 MB</td>
<td>0.84 MB</td>
</tr>
</tbody>
</table>

August 13, 2021

ListTagsForResource API Endpoint

We removed support for pagination in this endpoint; i.e., the maxResults request field and nextToken request/response field. (Pagination did not work correctly.)

August 10, 2021

Amazon IVS Player SDK 1.4.0

Platform | Downloads and Changes
----------|------------------------
All       | • Fixed a rare issue in which VOD playback could stall if a seek happens right after a DURATION_CHANGED event or READY state update.
          | • Fixed a memory leak when playing streams with ID3 metadata.
          | • Fixed an edge case in which injected captions could be rendered incorrectly.
          | • Improved the performance of the player's adaptive bitrate streaming algorithm.
          | • Improved player stability by reducing occurrences of rare crashes.
          | • Added a log warning message when the player is accessed from a different thread than it was created on.
          | • Updated getLiveLatency() documentation to be more specific about how latency is calculated, from the server to the player.

Web player 1.4.0 (p. 41) & Video.js integration (p. 53) & JW player integration (p. 62) | NPM Package: https://www.npmjs.com/package/amazon-ivs-player
Web player 1.4.0 (p. 41) & Video.js integration (p. 53) & JW player integration (p. 62) | Script asset: https://player.live-video.net/1.4.0/amazon-ivs-player.min.js
Web player 1.4.0 (p. 41) & Video.js integration (p. 53) & JW player integration (p. 62) | Video.js tech asset: https://player.live-video.net/1.4.0/amazon-ivs-videojs-tech.min.js
<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Reference documentation:</strong> <a href="https://aws.github.io/amazon-ivs-player-docs/1.4.0/web/">https://aws.github.io/amazon-ivs-player-docs/1.4.0/web/</a></td>
</tr>
</tbody>
</table>

- Fixed an edge case in which the `TIMED_METADATA` event did not fire on iOS Safari.
- Improved performance of the player’s adaptive bitrate streaming algorithm when playing low-latency streams on Firefox.
- Fixed documentation for `getDuration()`, which always returns Infinity for live streams.
- Fixed a bug in which autoplaying on desktop Safari sometimes failed.
- Fixed an error in which "Cannot read property 'collectLogs' of undefined" is reported in the developer console.
- **Video.js:** Added support for picture-in-picture mode.
- **Web:** Added a new method, `setRequestCredentials`. This controls whether the player makes credentialed requests to cross-origin endpoints. The remote endpoint needs to respond with the appropriate CORS response headers (like `Access-Control-Allow-Origin`, matching the request's Origin) and `Access-Control-Allow-Credentials` must be true.

This setting persists throughout the player instance's lifecycle. Therefore, all subsequent `player.load()` calls with URL endpoints should respond with appropriate CORS headers.

This method has no effect on iOS browser platforms. To allow credentialed cross-origin requests on iOS platforms, users must explicitly allow Cross-site Tracking and allow Cookies; these are in the settings for the device and the respective browser app.
Mobile SDK Size: Android

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Compressed Size</th>
<th>Uncompressed Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>arm64-v8a</td>
<td>1.00 MB</td>
<td>2.79 MB</td>
</tr>
<tr>
<td>armeabi-v7a</td>
<td>0.83 MB</td>
<td>2.15 MB</td>
</tr>
</tbody>
</table>
### Architecture

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Compressed Size</th>
<th>Uncompressed Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>x86_64</td>
<td>1.11 MB</td>
<td>3.06 MB</td>
</tr>
<tr>
<td>x86</td>
<td>1.11 MB</td>
<td>2.93 MB</td>
</tr>
</tbody>
</table>

### Mobile SDK Size: iOS

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Compressed Size</th>
<th>Uncompressed Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>universal</td>
<td>0.89 MB</td>
<td>1.91 MB</td>
</tr>
<tr>
<td>arm64</td>
<td>0.45 MB</td>
<td>1.05 MB</td>
</tr>
<tr>
<td>armv7</td>
<td>0.44 MB</td>
<td>0.84 MB</td>
</tr>
</tbody>
</table>

---

**July 27, 2021**

**Amazon IVS Broadcast SDK: Android 1.0.0 and iOS 1.0.0**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android Broadcast SDK 1.0.0 (p. 22)</td>
<td>Reference documentation: <a href="https://aws.github.io/amazon-ivs-broadcast-docs/1.0.0/android/">https://aws.github.io/amazon-ivs-broadcast-docs/1.0.0/android/</a></td>
</tr>
<tr>
<td>iOS Broadcast SDK 1.0.0 (p. 28)</td>
<td>Download: <a href="https://broadcast.live-video.net/1.0.0/AmazonIVSBroadcast.xcframework.zip">https://broadcast.live-video.net/1.0.0/AmazonIVSBroadcast.xcframework.zip</a> Reference documentation: <a href="https://aws.github.io/amazon-ivs-broadcast-docs/1.0.0/ios/">https://aws.github.io/amazon-ivs-broadcast-docs/1.0.0/ios/</a></td>
</tr>
</tbody>
</table>

---

**June 1, 2021**

**Amazon IVS Player SDK: Android 1.3.3 and iOS 1.3.3**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android and iOS</td>
<td>Fixed an issue where high-resolution portrait video was considered as not supported, although the device could support it.</td>
</tr>
<tr>
<td>Android player 1.3.3 (p. 44)</td>
<td>Reference documentation: <a href="https://aws.github.io/amazon-ivs-player-docs/1.3.3/android/">https://aws.github.io/amazon-ivs-player-docs/1.3.3/android/</a></td>
</tr>
</tbody>
</table>
May 19, 2021

Amazon IVS Player SDK: Android 1.3.2

**Reference documentation:** https://aws.github.io/amazon-ivs-player-docs/1.3.2/android/

To improve player stability, additional checks were implemented to ignore API calls after the `player.release()` method is called.

May 5, 2021

Amazon IVS Player SDK 1.3

<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
</table>
| All      | • Updated SDK documentation for using TextCue usage documentation. See the latest the Player SDK References on the Amazon IVS documentation landing page.  
• Fixed an issue with audio playback of malformed mono input streams.  
• Fixed a rare playback error that could occur when playing content outside of the live HLS window.  
• Improved the player's ability to play standard HLS live and recorded streams.  
• Improved the accuracy of `getLiveLatency`, notably ensuring it is reset to zero when loading a new stream.  
• Improved the ABR (adaptive bitrate streaming) algorithm to increase video quality more quickly when network connections improve.  
• Improved player stability by reducing occurrences of rare crashes. |
<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web player 1.3.1 (p. 41) &amp; Video.js integration (p. 53) &amp; JW player integration (p. 62)</td>
<td>NPM Package: <a href="https://www.npmjs.com/package/amazon-ivs-player">https://www.npmjs.com/package/amazon-ivs-player</a></td>
</tr>
<tr>
<td></td>
<td>Script asset: <a href="https://player.live-video.net/1.3.1/amazon-ivs-player.min.js">https://player.live-video.net/1.3.1/amazon-ivs-player.min.js</a></td>
</tr>
<tr>
<td></td>
<td>Video.js tech asset: <a href="https://player.live-video.net/1.3.1/amazon-ivs-videojs-tech.min.js">https://player.live-video.net/1.3.1/amazon-ivs-videojs-tech.min.js</a></td>
</tr>
<tr>
<td></td>
<td>Reference documentation: <a href="https://aws.github.io/amazon-ivs-player-docs/1.3.1/web/">https://aws.github.io/amazon-ivs-player-docs/1.3.1/web/</a></td>
</tr>
<tr>
<td></td>
<td>• Fixed an issue where seek calls executed immediately after load were sometimes ignored, causing the player to begin at the wrong position.</td>
</tr>
<tr>
<td></td>
<td>• Fixed several issues with seeking within recorded content (also known as VOD).</td>
</tr>
<tr>
<td></td>
<td>• Fixed an issue where playback could fail in suboptimal network conditions.</td>
</tr>
<tr>
<td></td>
<td>• Added support for IVS Timed Metadata on iOS mobile web browsers.</td>
</tr>
<tr>
<td></td>
<td>• Fixed a bug where autoplaying in desktop Safari sometimes failed.</td>
</tr>
<tr>
<td></td>
<td>• The Web SDK <code>getVersion</code> function no longer appends a hash to the player version.</td>
</tr>
<tr>
<td></td>
<td>• Fixed an issue where seeking to the exact start of a buffered range may result in another seek forward.</td>
</tr>
<tr>
<td></td>
<td>• Enabled low-latency ABR (adaptive bitrate streaming) in macOS Safari 14 and later.</td>
</tr>
<tr>
<td></td>
<td>• Fixed an issue with loading the player in a server context, by removing an unsafe import side effect.</td>
</tr>
<tr>
<td></td>
<td>• Changed the <code>amazon-ivs-player</code> NPM package so it exports the LogLevel enum, which is used by <code>setLogLevel</code>.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The Web Player 1.3.0 NPM package exists but does not work. It is marked as deprecated on NPM. Use Web Player 1.3.1 or newer, as documented.</td>
</tr>
<tr>
<td>Platform</td>
<td>Downloads and Changes</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Android player 1.3.0 (p. 44)</td>
<td>Reference documentation: <a href="https://aws.github.io/amazon-ivs-player-docs/1.3.0/android/">https://aws.github.io/amazon-ivs-player-docs/1.3.0/android/</a></td>
</tr>
<tr>
<td></td>
<td>• Fixed an issue where the player SDK crashed if the app targeted Android 11 (API level 30) and the user was running Android 11 on a cellular network.</td>
</tr>
<tr>
<td></td>
<td>• Fixed a network recovery issue. Playback is now automatically paused when the network connection is lost, and it is resumed when the connection is restored. Use the onNetworkUnavailable callback in Player.Listener to observe network state changes.</td>
</tr>
<tr>
<td></td>
<td>• Fixed an issue where player controls could not be hidden with setControlsEnabled(false) while playing VODs.</td>
</tr>
<tr>
<td></td>
<td>• Fixed an issue where the SDK could crash if the client app uses an old (pre-4.0) version of OkHttp.</td>
</tr>
<tr>
<td></td>
<td>• The Amazon IVS Android player library moved from a JCenter repository to Maven Central.</td>
</tr>
<tr>
<td></td>
<td>• Removed BuildConfig version properties from the library.</td>
</tr>
</tbody>
</table>
## Downloads and Changes

<table>
<thead>
<tr>
<th>Platform</th>
<th>Download: <a href="https://player.live-video.net/1.3.0/AmazonIVSPlayer.xcframework.zip">https://player.live-video.net/1.3.0/AmazonIVSPlayer.xcframework.zip</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS Player 1.3.0 (p. 48)</td>
<td><strong>Reference documentation:</strong> <a href="https://aws.github.io/amazon-ivs-player-docs/1.3.0/ios/">https://aws.github.io/amazon-ivs-player-docs/1.3.0/ios/</a></td>
</tr>
</tbody>
</table>

- Fixed an issue where if there was an audio sample-rate change within a single HLS media segment, the iOS SDK could not handle it properly. This could result in an unexpected memory increase and playback failure or a crash, due to bad media.
- Fixed a network-recovery issue. Playback is now automatically paused when the network connection is lost, and it is resumed when the connection is restored. Use the `playerNetworkDidBecomeUnavailable` delegate method to observe network state changes.
- Fixed an issue which caused an iOS memory increase that could happen over time.
- Added graceful handling of audio hardware problems. Playback is now automatically paused in the event of a media-services reset notification (`AVAudioSessionMediaServicesWereResetNotification`). Note that a playback error may still occur if media is playing when the reset occurs.
- Added audio-session interruption handling. Playback is now automatically paused when an audio-session interruption begins. When the interruption ends, playback automatically resumes if the player was previously playing and the interruption options indicate that the app should resume playback.

## April 26, 2021

**Service Quotas Integration with CloudWatch Usage Metrics**

You can use CloudWatch to proactively manage your service quotas, via CloudWatch usage metrics. See Amazon IVS Service Quotas.
April 13, 2021

New CloudWatch Metrics

CloudWatch metrics were added for concurrent views and concurrent streams. See Monitoring Amazon IVS with Amazon CloudWatch (p. 93).

The names of related service quotas were updated to match the new metrics. See Amazon IVS Service Quotas (p. 111).

For a full definition of "view," see the Amazon IVS Glossary (p. 124).

April 7, 2021

Auto-Record to S3 (R2S3)

Amazon IVS now enables you to save your live video content to Amazon S3. Saved video is available later for actions like editing or replaying as a VOD.

When you enable recording for a channel, all live broadcasts of the channel are stored to an S3 bucket of your choice. All available quality renditions and thumbnails images are saved. Your recording configuration also is saved, so it can be easily re-used for additional channels.

You can set up a recording configuration and enable/disable recording through the Amazon IVS console or API. For details, see Getting Started with Amazon IVS (p. 3) and the Amazon IVS API Reference.

January 28, 2021

Amazon IVS Player SDK: JW Player Integration 1.2.0

The Amazon IVS player now integrates with JW Player. See Amazon IVS Player: JW Player Integration (p. 62).

Known Issue: In some cases, the duration of the video appears to be 00:00 and the playhead does not seek if dragged on the seekbar. This happens only when watching an ad-free playlist with a mixture of Amazon IVS live streams and VODs, using Safari on an iPhone.

December 16, 2020

Amazon IVS Player: SDK for Android 1.2.1

Reference documentation: https://aws.github.io/amazon-ivs-player-docs/1.2.1/android/

This release includes an Android Player patch which fixes an issue: in prior Android player SDK releases, the SDK crashes if the app targets Android 11 (API level 30) and the user is running Android 11 on a cellular network.
## Amazon IVS Player SDK 1.2.0

<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Improved detection of Amazon IVS streams, so metrics are more accurate.</td>
</tr>
<tr>
<td>Web player 1.2.0 (p. 41) &amp; Video.js integration (p. 53)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NPM Package: <a href="https://www.npmjs.com/package/amazon-ivs-player">https://www.npmjs.com/package/amazon-ivs-player</a></td>
</tr>
<tr>
<td></td>
<td>Script asset: <a href="https://player.live-video.net/1.2.0/amazon-ivs-player.min.js">https://player.live-video.net/1.2.0/amazon-ivs-player.min.js</a></td>
</tr>
<tr>
<td></td>
<td>Video.js tech asset: <a href="https://player.live-video.net/1.2.0/amazon-ivs-videojs-tech.min.js">https://player.live-video.net/1.2.0/amazon-ivs-videojs-tech.min.js</a></td>
</tr>
<tr>
<td></td>
<td>Reference documentation: <a href="https://aws.github.io/amazon-ivs-player-docs/1.2.0/web/">https://aws.github.io/amazon-ivs-player-docs/1.2.0/web/</a></td>
</tr>
<tr>
<td></td>
<td>• If the master playlist for a stream is unavailable, we now emit <code>ErrorNotAvailable</code> for all web playback sources.</td>
</tr>
<tr>
<td></td>
<td>• Updated reference documentation with respect to errors related to reaching the concurrent-viewers (CCV) limit.</td>
</tr>
<tr>
<td>Android player 1.2.0 (p. 44)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reference documentation: <a href="https://aws.github.io/amazon-ivs-player-docs/1.2.0/android/">https://aws.github.io/amazon-ivs-player-docs/1.2.0/android/</a></td>
</tr>
<tr>
<td></td>
<td>• Fixed an issue where the <code>getSessionId</code> function crashed on Android.</td>
</tr>
<tr>
<td></td>
<td>• Updated reference documentation with respect to errors related to reaching the concurrent-viewers (CCV) limit.</td>
</tr>
<tr>
<td></td>
<td><strong>Known Issue:</strong> The player SDK will crash if the app targets Android 11 (API level 30) and the user is running Android 11 on a cellular network. This will be fixed in the next release. In the meantime, we recommend targeting a previous Android API level (29 or lower).</td>
</tr>
<tr>
<td>iOS Player 1.2.0 (p. 48)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Download: <a href="https://player.live-video.net/1.2.0/AmazonIVSPlayer.xcframework.zip">https://player.live-video.net/1.2.0/AmazonIVSPlayer.xcframework.zip</a></td>
</tr>
<tr>
<td></td>
<td>Reference documentation: <a href="https://aws.github.io/amazon-ivs-player-docs/1.2.0/ios/">https://aws.github.io/amazon-ivs-player-docs/1.2.0/ios/</a></td>
</tr>
<tr>
<td></td>
<td>• Fixes a potential source of memory corruption when switching the stream URL or closing the player.</td>
</tr>
</tbody>
</table>
|          | • Resolves an issue that could cause playback to fail when the best audio pitch correction could
November 12, 2020

New Event Field, stream_id

The stream_id field was added to several events. See Using Amazon EventBridge with Amazon IVS (p. 85).

November 9, 2020

Add Metadata Viewing to Console

Timed metadata can now be viewed from the Amazon IVS console. In the Amazon IVS User Guide, see the new section on Viewing Timed Metadata (p. 68) in Embedding Metadata within a Video Stream.

October 30, 2020

CloudFormation Support

Amazon IVS now supports AWS CloudFormation. This enables Amazon IVS customers to create and manage channels, stream keys, and playback key pairs with AWS CloudFormation.

Amazon IVS support for CloudFormation is available in all AWS regions where Amazon IVS is available. To get started, see the Amazon IVS product page or the Amazon IVS information in the AWS CloudFormation User Guide.

October 27, 2020

Higher Limits for Channels, CCV, and CCB

We increased three service-quota limits:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>not be enabled before starting playback. Pitch correction improves audio quality at playback speeds faster or slower than normal. If pitch correction cannot be enabled or the highest quality correction algorithm is unavailable, a message is logged but playback continues.</td>
</tr>
<tr>
<td></td>
<td><strong>Known Issue</strong>: If there is an audio sample-rate change within a single HLS media segment, the iOS SDK cannot handle it properly. This can result in an unexpected memory increase and playback failure or a crash, due to bad media. This will be fixed in the next major iOS player release.</td>
</tr>
</tbody>
</table>
• The maximum number of *channels* that users can create, per AWS region, increased from 500 to 5,000.
• The maximum number of *concurrent viewers* allowed to play back a live channel, across all channels in an AWS region, increased from 3,000 to 15,000.
• The maximum number of *concurrent broadcasts* (channels that can be streamed simultaneously), per AWS region, increased from 30 to 100.

These increases are available in all regions where Amazon IVS is available. To learn more, see Amazon IVS Service Quotas (p. 111) in the Amazon IVS User Guide.

October 9, 2020

**New Service Quotas and EventBridge Event**

There are now service quotas and EventBridge events related to ingest resolution. See Amazon IVS Service Quotas (p. 111) and Using Amazon EventBridge with Amazon IVS (p. 85).

**Amazon IVS Player: SDK for Web 1.1.2**

**NPM Package:** [https://www.npmjs.com/package/amazon-ivs-player](https://www.npmjs.com/package/amazon-ivs-player)

**Script asset:** [https://player.live-video.net/1.1.2/amazon-ivs-player.min.js](https://player.live-video.net/1.1.2/amazon-ivs-player.min.js)

**Video.js tech asset:** [https://player.live-video.net/1.1.2/amazon-ivs-videojs-tech.min.js](https://player.live-video.net/1.1.2/amazon-ivs-videojs-tech.min.js)

**Reference documentation:** [https://aws.github.io/amazon-ivs-player-docs/1.1.2/web/](https://aws.github.io/amazon-ivs-player-docs/1.1.2/web/)

This release includes a Web Player patch which fixes an issue that affected viewers using Microsoft Edge. For those viewers, if auto-quality mode is turned on for the stream (i.e., ABR is in effect), low-latency playback does not work; under these circumstances, streams played back with higher latency.

October 7, 2020

**Amazon IVS Player SDK 1.1.0**

The Amazon Interactive Video Service (IVS) Player SDKs use *semantic versioning*.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
</table>
| All      | • Fixed an issue where the player's adaptive bitrate algorithm could incorrectly drop quality to 160p.  
          | • The player now throws an error if there are no playable video qualities.  
          | • Updated VOD seek behavior: when attempting to seek beyond the end, the player seeks to the end instead of returning an error.  
<pre><code>      | • The player now throws a fatal error after exhausting all available qualities during error recovery. |
</code></pre>
<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Player 1.1.0</td>
<td><strong>NPM Package:</strong> <a href="https://www.npmjs.com/package/amazon-ivs-player">https://www.npmjs.com/package/amazon-ivs-player</a></td>
</tr>
<tr>
<td></td>
<td><strong>Script asset:</strong> <a href="https://player.live-video.net/1.1.0/amazon-ivs-player.min.js">https://player.live-video.net/1.1.0/amazon-ivs-player.min.js</a></td>
</tr>
<tr>
<td></td>
<td><strong>Video.js tech asset:</strong> <a href="https://player.live-video.net/1.1.0/amazon-ivs-videojs-tech.min.js">https://player.live-video.net/1.1.0/amazon-ivs-videojs-tech.min.js</a></td>
</tr>
<tr>
<td></td>
<td><strong>Reference documentation:</strong> <a href="https://aws.github.io/amazon-ivs-player-docs/1.1.0/web/">https://aws.github.io/amazon-ivs-player-docs/1.1.0/web/</a></td>
</tr>
<tr>
<td></td>
<td><strong>Known Issues:</strong></td>
</tr>
<tr>
<td></td>
<td>• If Video.js is not available, registerIVSQualityPlugin now throws an exception instead of writing to</td>
</tr>
<tr>
<td></td>
<td>console.error.</td>
</tr>
<tr>
<td></td>
<td>• If registerIVSTech or registerIVSQualityPlugin is called more than once, calls after the first one</td>
</tr>
<tr>
<td></td>
<td>now do nothing (instead of attempting to re-register).</td>
</tr>
<tr>
<td></td>
<td>• The type of the first parameter to registerIVSQualityPlugin has changed from VideoJS to any.</td>
</tr>
<tr>
<td></td>
<td>• Removed dependencies on browser context to enable server-side rendering.</td>
</tr>
<tr>
<td></td>
<td>• If the browser autopauses in response to unmuting, the player now fires the AUDIO_BLOCKED event and</td>
</tr>
<tr>
<td></td>
<td>resumes muted playback.</td>
</tr>
<tr>
<td></td>
<td>• Added network connectivity recovery. A network timeout will not result in an error state being</td>
</tr>
<tr>
<td></td>
<td>sent to the client app. Instead, when network connectivity is lost:</td>
</tr>
<tr>
<td></td>
<td>• If the app is playing, the player library sends the NETWORK_UNAVAILABLE event to the app and the</td>
</tr>
<tr>
<td></td>
<td>player enters the IDLE state. When connectivity is restored, the player library resumes playing</td>
</tr>
<tr>
<td></td>
<td>and the app receives a PLAYING event.</td>
</tr>
<tr>
<td></td>
<td>• If the app is paused, the NETWORK_UNAVAILABLE event is not sent to the app and the player library</td>
</tr>
<tr>
<td></td>
<td>remains in the IDLE state. When connectivity is restored, the player library stays in the IDLE</td>
</tr>
<tr>
<td></td>
<td>state.</td>
</tr>
<tr>
<td></td>
<td>• At any time, if the app tries to play, the player library attempts a normal play. The</td>
</tr>
<tr>
<td></td>
<td>NETWORK_UNAVAILABLE event is sent to the app and the player enters the IDLE state.</td>
</tr>
</tbody>
</table>
## Platform

<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android Player 1.1.0</td>
<td>Reference documentation: <a href="https://aws.github.io/amazon-ivs-player-docs/1.1.0/android/">https://aws.github.io/amazon-ivs-player-docs/1.1.0/android/</a></td>
</tr>
<tr>
<td></td>
<td>Known Issue: The player SDK will crash if the app targets Android 11 (API level 30) and the user is running Android 11 on a cellular network. This will be fixed in the next release. In the meantime, we recommend targeting a previous Android API level (29 or lower).</td>
</tr>
<tr>
<td>iOS Player 1.1.0</td>
<td>Download: <a href="https://player.live-video.net/1.1.0/AmazonIVSPlayer.xcframework.zip">https://player.live-video.net/1.1.0/AmazonIVSPlayer.xcframework.zip</a></td>
</tr>
<tr>
<td></td>
<td>Reference documentation: <a href="https://aws.github.io/amazon-ivs-player-docs/1.1.0/ios/">https://aws.github.io/amazon-ivs-player-docs/1.1.0/ios/</a></td>
</tr>
<tr>
<td></td>
<td>• Fixed an issue that could cause crashes, with this message from UIKit: &quot;Modifications to the layout engine must not be performed from a background thread after it has been accessed from the main thread.&quot; This could occur when backgrounding and foregrounding the application.</td>
</tr>
</tbody>
</table>

### September 14, 2020

**New Event Field, channel_name**

The channel_name field was added to several events. See Using Amazon EventBridge with Amazon IVS (p. 85).

### August 19, 2020

**Playback Authorization (Private Channels)**

Amazon IVS now offers customers the ability to create private channels, allowing customers to restrict which viewers can watch their streams. Customers control access to video playback by enabling playback authorization on channels and generating signed JSON Web Tokens (JWTs) for authorized playback requests. For details, see Setting Up Private Channels (p. 69).

A new authorized field in the Channel object indicates whether the channel is private. See the Amazon IVS API Reference.

### August 11, 2020

**Amazon IVS Player: SDK for iOS 1.0.6**

Download: <deprecated>
This release includes an iOS Player patch which fixes an issue that had prevented some iOS Player apps from being added to the Apple App Store. Specifically, apps built with bitcode enabled would fail App Store Connect validation after uploading.

August 5, 2020

Using Amazon EventBridge with Amazon IVS

Amazon IVS EventBridge events are now available through the Amazon EventBridge console. See the section on Creating Amazon EventBridge Rules for Amazon IVS (p. 86) in Using Amazon EventBridge with Amazon IVS, in the Amazon IVS User Guide.

July 15, 2020

Player Version 1.0

The Amazon Interactive Video Service (IVS) Player SDKs use semantic versioning.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Downloads and Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td><strong>Known Issue:</strong> For the <code>setAutoMaxQuality</code> and <code>setQuality</code> functions, the quality you provide is applied correctly to the current stream but is not applied correctly if you load a new stream. To avoid this, If you load a new stream, call this with a quality for the new stream after <code>PlayerState.READY</code>.</td>
</tr>
<tr>
<td>Web Player 1.0.0</td>
<td><strong>NPM Package:</strong> <a href="https://www.npmjs.com/package/amazon-ivs-player">https://www.npmjs.com/package/amazon-ivs-player</a></td>
</tr>
<tr>
<td></td>
<td><strong>Script asset:</strong> <a href="https://player.live-video.net/1.0.0/amazon-ivs-player.min.js">https://player.live-video.net/1.0.0/amazon-ivs-player.min.js</a></td>
</tr>
<tr>
<td></td>
<td><strong>Video.js tech asset:</strong> <a href="https://player.live-video.net/1.0.0/amazon-ivs-videojs-tech.min.js">https://player.live-video.net/1.0.0/amazon-ivs-videojs-tech.min.js</a></td>
</tr>
<tr>
<td></td>
<td><strong>Reference documentation:</strong> <a href="https://aws.github.io/amazon-ivs-player-docs/1.0.0/web/">https://aws.github.io/amazon-ivs-player-docs/1.0.0/web/</a></td>
</tr>
<tr>
<td></td>
<td><strong>Known Issues:</strong></td>
</tr>
<tr>
<td></td>
<td>• When playing a VOD on an iOS mobile browser (e.g. Safari or Chrome), seeking backwards will mute the player. To avoid this, call <code>player.setMuted(false)</code> after seeking.</td>
</tr>
<tr>
<td></td>
<td>• When playing a VOD on an iOS mobile browser, seeking backwards works intermittently when directly selecting the desired position. To avoid this, drag the seek bar to the desired position.</td>
</tr>
<tr>
<td>Platform</td>
<td>Downloads and Changes</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>• When playing a VOD on an iOS mobile browser using the Video.js integration, the replay button does not work properly. To avoid this, hide the replay button when initializing Video.js: <a href="https://docs.videojs.com/tutorial-components.html#play-toggle">https://docs.videojs.com/tutorial-components.html#play-toggle</a>.</td>
</tr>
<tr>
<td>Android Player 1.0.0</td>
<td><strong>Reference documentation:</strong> <a href="https://aws.github.io/amazon-ivs-player-docs/1.0.0/android/">https://aws.github.io/amazon-ivs-player-docs/1.0.0/android/</a></td>
</tr>
<tr>
<td></td>
<td><strong>Known Issue:</strong> Backgrounding and foregrounding can cause audio/video de-synchronization for VOD playback on Android.</td>
</tr>
<tr>
<td>iOS Player 1.0.0</td>
<td><strong>Download:</strong> &lt;deprecated&gt;</td>
</tr>
<tr>
<td></td>
<td><strong>Reference documentation:</strong> <a href="https://aws.github.io/amazon-ivs-player-docs/1.0.0/ios/">https://aws.github.io/amazon-ivs-player-docs/1.0.0/ios/</a></td>
</tr>
<tr>
<td></td>
<td><strong>Known Issues:</strong></td>
</tr>
<tr>
<td></td>
<td>• Backgrounding and foregrounding cause live and VOD playback failure. To avoid this, pause the stream when the UIApplicationDidEnterBackgroundNotification is received and resume play on the UIApplicationDidBecomeActiveNotification.</td>
</tr>
<tr>
<td></td>
<td>• iOS 10 devices may experience a crash when returning from background. To avoid this, set the layer's <code>player</code> property to <code>nil</code> before entering the background.</td>
</tr>
</tbody>
</table>