Developer Guide

Amazon Chime SDK

Amazon Chime SDK: Developer Guide

Table of Contents

What is the Amazon Chime SDK?

The Amazon Chime SDK is a set of real-time communications components that you can use to quickly add messaging, audio, video, and screen sharing capabilities to your web or mobile applications.

You can use the Amazon Chime SDK to build real-time media applications that can send and receive audio and video and allow content sharing. For detailed information about Amazon Chime SDK API actions, see the Amazon Chime SDK API Reference.

Pricing

The Amazon Chime SDK offers pay-for-use pricing with no upfront fees. You can choose to implement some or all of the available media modalities (audio, video, and screen share) for a single rate. Messaging, media pipelines, speech enhancement, and PSTN audio capabilities are also available with pay-for-use pricing. For more information, see Amazon Chime SDK pricing.

Resources

The following related resources can help you as you work with this service.

- Classes & Workshops – Links to role-based and specialty courses, in addition to self-paced labs to help sharpen your AWS skills and gain practical experience.
- AWS Developer Center – Explore tutorials, download tools, and learn about AWS developer events.
- AWS Developer Tools – Links to developer tools, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- Getting Started Resource Center – Learn how to set up your AWS account, join the AWS community, and launch your first application.
- Hands-On Tutorials – Follow step-by-step tutorials to launch your first application on AWS.
- AWS Whitepapers – Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts.
- AWS Support Center – The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.

- AWS Support – The primary webpage for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.
- Contact Us – A central contact point for inquiries concerning AWS billing, account, events, abuse, and other issues.
- AWS Site Terms – Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

Using the Amazon Chime SDK

You use the Amazon Chime SDK to build real-time media applications that can send and receive audio and video and allow content sharing. The Amazon Chime SDK works independently of any Amazon Chime administrator accounts, and it does not affect meetings hosted on Amazon Chime. Instead, the Amazon Chime SDK provides builder tools that you use to build your own meeting applications.

Topics

Prerequisites to use the Amazon Chime SDK

Using the Amazon Chime SDK requires the following:

- The ability to program.
- An AWS account.
- An IAM role with a policy that grants permission to access Amazon Chime API actions used by the Amazon Chime SDK, such as the AWS managed **AmazonChimeSDK** policy. For more information, see How Amazon Chime works with IAM and Allow users to access Amazon Chime SDK actions in the *Amazon Chime SDK Administrator Guide*.
- For the majority of use cases, you also need the following:

- A **server application** – Manages meeting and attendee resources, and serves those resources to the client application. The server application is created in the AWS account and must have access to the IAM role mentioned previously.
- A **client application** – Receives meeting and attendee information from the server application, and uses that information to make media connections.

Concepts for the Amazon Chime SDK

The following terminology and concepts are central to understanding how to use the Amazon Chime SDK.

meeting

An ephemeral resource identified by a unique `MeetingId`. The `MeetingId` is placed onto a group of media services that host the active meeting.

media service group

The group of media services that hosts an active meeting.

media placement

A set of regionalized URLs that represents a media service group. Attendees connect to the media service group with their clients to send and receive real-time audio and video, and share their screens.

attendee

A meeting participant that is identified by a unique `AttendeeId`. Attendees may freely join and leave meetings using a client application built with an Amazon Chime SDK client library.

join token

A unique token assigned to each attendee. Attendees use the join token to authenticate with the media service group.

Architecture for the Amazon Chime SDK

The following list describes how the different components of the Amazon Chime SDK architecture work together to support meetings and attendees, audio, video, and content sharing.

Meetings and attendees

When the server application creates an Amazon Chime SDK meeting, the meeting is assigned to a region-specific media service. The hosts in the service are responsible for securely transferring real-time media between attendee clients. Each created attendee is assigned a unique join token, an opaque secret key that your server application must securely transfer to the client authorized to join the meeting on behalf of an attendee. Each client uses a join token to authenticate with the media service group. Clients use a combination of secure WebSockets and Datagram Transport Layer Security (DTLS) to securely signal the media service group, and to send and receive media to and from other attendees through the media service group.

Audio

The media service mixes audio together from each attendee and sends the mix to each recipient, after subtracting their own audio from the mix. The Amazon Chime SDKs sample audio at the highest rate supported by the device and browser, up to a maximum of 48kHz. We use the Opus codec to encode audio, with a default bitrate of 32kbps, which can be increased to up to 128kbps stereo and 64kbps mono.

Video

The media service acts as a Selective Forwarding Unit (SFU) using a publish and subscribe model. Each attendee can publish one video source, up to a total of 25 simultaneous videos per meeting. The Amazon Chime SDK client library for JavaScript supports video resolutions up to 1280x720 at 30 frames per second without simulcast, and 15 frames per second with simulcast. The Amazon Chime SDK client libraries for iOS, Android, and Windows support video resolutions up to 1280x720 and 30 frames per second, however the actual framerate and resolution is automatically managed by the Amazon Chime SDK.

When active, video simulcast sends each video stream in two different resolutions and bitrates. Clients with bandwidth constraints automatically subscribe to the lower bitrate stream. Video encoding and decoding uses hardware acceleration where available to improve performance.

Data messages

In addition to audio and video content, meeting attendees can send each other real-time data messages of up to 2 KB each. You can use messages to implement custom meeting features such as whiteboarding, chat, real-time emoji reactions, and application-specific floor control signaling.

Content sharing

The client application can share audio and video content, such as screen captures or media files. Content sharing supports pre-recorded content video up to 1280x720 at 15 frames per second, and audio up to 48kHz at 64kbps. Screen capture for content sharing is supported up to 15 frames per second, but may be limited by the capabilities of the device and browser.

Quotas for the Amazon Chime SDK

Quotas for the Amazon Chime SDK service are documented in the AWS General Reference. For more information, see Amazon Chime SDK endpoints and quotas in the *AWS General Reference*.

> **Note**
>
> Service quotas are per API endpoint. When requesting a service quota increase, be sure to request the increase on all the API endpoints that your application uses.

Amazon Chime SDK system requirements

The following system requirements apply to applications created with the Amazon Chime SDK.

Supported browsers, Amazon Chime SDK client library for JavaScript

Operating system	Browser	Supported versions	Notes
Windows	Mozilla Firefox	75 and later	
	Google Chrome	78 and later	
	Chromium-based Edge	79 and later	
	Chromium-based Electron	7 and later	With Chrome version 78 and later.
	Opera	66 and later	
macOS	Mozilla Firefox	75 and later	

Operating system	Browser	Supported versions	Notes
	Google Chrome	78 and later	
	Chromium-based Edge	79 and later	
	Chromium-based Electron		
	Safari	13 and later	
	Opera	66 and later	
iOS	Mozilla Firefox	10 and later	Audio and video only, no content sharing.
	Google Chrome	78 and later	Audio and video only, no content sharing.
	Safari	13 and later	Audio and video only, no content sharing.
	WKWebView	14.3 and later	Audio and video only, no content sharing.
Android	Google Chrome	10 and later	Audio and video only, no content sharing.
	Samsung	12 and later	Audio and video only, no content sharing.
	Chromium WebView	5 and later	Audio and video only, no content sharing.
Ubuntu LTS 16.04 and later	Google Chrome	78 and later	

Amazon Chime SDK client library for iOS

- iOS version 13 and later

Amazon Chime SDK client library for Android

- Android OS version 5 and later, ARM and ARM64 architecture

Available AWS Regions for the Amazon Chime SDK service

The following tables list the features of the Amazon Chime SDK service and the AWS Regions that provide each service.

> **Note**
>
> Regions marked with an asterisk (*) must be enabled in your AWS account. AWS blocks those Regions by default. For more information about enabling Regions, see Specify which AWS Regions your account can use, in the *AWS Account Management Reference*.

Topics

Console Regions

You use the Amazon Chime SDK console to configure resources and learn more about the Amazon Chime SDK service.

AWS Region	Console
Asia Pacific (Seoul)	Yes
Asia Pacific (Singapore)	
Asia Pacific (Sydney)	Yes
Asia Pacific (Tokyo) (ap-northeast-1)	Yes
Canada (Central) (ca-central-1)	Yes
Europe (Frankfurt) (eu-central-1)	Yes
Europe (Ireland) (eu-west-1)	Yes
Europe (London) (eu-west-2)	Yes
US East (N. Virginia) (us-east-1)	Yes
US West (Oregon) (us-west-2)	Yes

Call analytics Regions

The following table lists the AWS Regions available for analytics, transcription, and call recording.

AWS Region	Voice analytics	Transcription	Call recording
US East (N. Virginia) (us-east-1)	Yes	Yes	Yes
US West (Oregon) (us-west-2)	Yes	Yes	Yes
Europe (Frankfurt) (eu-central-1)	No	Yes	Yes

Meeting Regions

Amazon Chime SDK meetings have *control Regions* and *media Regions*. A control Region provides the API endpoint used to create, update and delete meetings. Control Regions also receive and process Understanding Amazon Chime SDK meeting lifecycle events.

Media Regions host the actual meetings, and clients connect to your media Regions. You specify the media Region when you call the CreateMeeting API.

A control Region can create a meeting in any media Region in the same AWS partition. However, you can only update a meeting in the control Region used to create the meeting.

For more information about selecting control and media Regions, see Using meeting Regions.

The following table lists the Regions that provide control, media, or both.

AWS Region	Meeting control	Meeting media
Africa (Cape Town) (af-south-1)*	Yes**	Yes
Asia Pacific (Mumbai) (ap-south-1)	Yes	Yes
Asia Pacific (Seoul) (ap-northeast-2)	Yes	Yes
Asia Pacific (Singapore) (ap-southeast-1)	Yes	Yes
Asia Pacific (Sydney) (ap-southeast-2)	Yes	Yes
Asia Pacific (Tokyo) (ap-northeast-1)	Yes	Yes
Canada (Central) (ca-central-1)	Yes	Yes

AWS Region	Meeting control	Meeting media
Europe (Frankfurt) (eu-central-1)	Yes	Yes
Europe (Ireland) (eu-west-1)		Yes
Europe (London) (eu-west-2)	Yes	Yes
Europe (Milan) (eu-south-1)*		Yes
Europe (Paris) (eu-west-3)		Yes
Europe (Stockholm) (eu-north-1)		Yes
Israel (Tel Aviv) (il-central-1)*	Yes**	Yes
South America (São Paulo) (sa-east-1)		Yes
US East (Ohio) (us-east-2)		Yes
US East (N. Virginia) (us-east-1)	Yes	Yes
US West (N. California) (us-west-1)		Yes
US West (Oregon) (us-west-2)	Yes	Yes
AWS GovCloud (US-East) (us-gov-east-1)	Yes	Yes
AWS GovCloud (US-West) (us-gov-west-1)	Yes	Yes

*You must enable these Regions in your AWS account. For more information, refer to Enable a Region in the *AWS General Reference*.

**Meetings that use meeting control in this Region can only host media in this Region.

> **Note**
>
> To create a meeting in an AWS GovCloud (US) Region, you must use a control Region in GovCloud. Also, control Regions in GovCloud can only make meetings in AWS GovCloud (US) Regions.

Media pipeline Regions

Amazon Chime SDK media pipelines have *control Regions* and *media Regions*. A control Region provides the media pipeline API endpoint used to create and delete media pipelines. You also use control Regions to receive and process media pipeline events.

Media Regions run your media pipelines, and the system automatically selects the same media Region as the meeting.

You can use a control Region to create a media pipeline in any data Region. The media pipeline can join a meeting in any meeting media Region.

AWS Region	Control	Media
Africa (Cape Town) (af-south-1)*		Yes
Asia Pacific (Mumbai) (ap-south-1)	Yes	Yes
Asia Pacific (Seoul) (ap-northeast-2)	Yes	Yes
Asia Pacific (Singapore) (ap-southeast-1)	Yes	Yes
Asia Pacific (Sydney) (ap-southeast-2)	Yes	Yes

AWS Region	Control	Media
Asia Pacific (Tokyo) (ap-northeast-1)	Yes	Yes
Canada (Central) (ca-central-1)	Yes	Yes
Europe (Frankfurt) (eu-central-1)	Yes	Yes
Europe (Ireland) (eu-west-1)		Yes
Europe (London) (eu-west-2)	Yes	Yes
Europe (Milan) (eu-south-1)*		Yes
Europe (Paris) (eu-west-3)		Yes
Europe (Stockholm) (eu-north-1)		Yes
South America (São Paulo) (sa-east-1)		Yes
US East (Ohio) (us-east-2)		Yes
US East (N. Virginia) (us-east-1)	Yes	Yes
US West (N. California) (us-west-1)		Yes
US West (Oregon) (us-west-2)	Yes	Yes

*You must enable these Regions in your AWS account. For more information, refer to Enable a Region in the *AWS General Reference*.

Messaging Regions

Amazon Chime SDK messaging has *control regions* and *data regions*. The control Region exposes the messaging API endpoint, and the data Region stores the messages. If you use Amazon Kinesis to stream messaging data, or AWS Lambda functions for channel flows, they should reside in the control Region.

AWS Region	Control	Data
Europe (Frankfurt) (eu-central-1)	Yes	Yes
US East (N. Virginia) (us-east-1)	Yes	Yes

Voice Regions

Amazon Chime SDK SIP (Session Initiation Protocol) features have *API regions* and *media regions*, and *PSTN regions*. The API regions provide the API endpoints for creating and configuring SIP features. The media Regions contain Amazon Chime SDK Voice Connectors and SIP media applications. The PSTN Regions enable customers to connect on-premises phone systems to the public telephone network. Additionally, PSTN Regions support phone number provisioning and management.

AWS Region	API	Media	PSTN
Asia Pacific (Seoul) (ap-northeast-2)	Yes	Yes	
Asia Pacific (Singapore) (ap-southeast-1)	Yes	Yes	
Asia Pacific (Sydney) (ap-southeast-2)	Yes	Yes	
Asia Pacific (Tokyo) (ap-northeast-1)	Yes	Yes	

AWS Region	API	Media	PSTN
Canada (Central) (ca-central-1)	Yes	Yes	
Europe (Frankfurt) (eu-central-1)	Yes	Yes	
Europe (Ireland) (eu-west-1)	Yes	Yes	
Europe (London) (eu-west-2)	Yes	Yes	
US East (N. Virginia) (us-east-1)	Yes	Yes	Yes*
US West (Oregon) (us-west-2)	Yes	Yes	Yes*

*See the Amazon Chime SDK Pricing page for information about the availability of phone numbers in specific AWS regions.

Learn about the Amazon Chime SDK client libraries

Before you can build real-time meeting clients with the Amazon Chime SDK, you must integrate your client application with an Amazon Chime SDK client library. The following client libraries are available:

- Amazon Chime SDK client library for Android – A Kotlin library that helps you build Amazon Chime SDK applications on supported Android devices.
- Amazon Chime SDK signaling client library for C++ – A C++ library that helps you set up signaling connections to Amazon Chime SDK meetings on embedded devices.
- Amazon Chime SDK client library for iOS – A Swift library that helps you build Amazon Chime SDK applications on supported iOS devices.
- Amazon Chime SDK client library for JavaScript (NPM) – A JavaScript library with TypeScript type definitions that helps you build Amazon Chime SDK applications in WebRTC-enabled browsers.

- Amazon Chime SDK client library for Windows. A C++ library that helps you build Amazon Chime SDK applications on supported devices.

To learn how to integrate your client application with the Amazon Chime SDK, see the actions in the client library `README.md` files. Use the demos to learn how to build specific media components for your application.

Understanding SIP integration using an Amazon Chime SDK Voice Connector

Integrate your SIP-compatible voice infrastructure with an Amazon Chime SDK Voice Connector to make SIP voice calls. You must use the `us-east-1` or `us-east-2` Regions. You must have an IP Private Branch Exchange (PBX), Session Border Controller (SBC), or other voice infrastructure with internet access that supports Session Initiation Protocol (SIP). For more information, see Before you begin in the *Amazon Chime SDK Administrator Guide*.

To integrate your voice infrastructure

1. Create an Amazon Chime SDK Voice Connector under your AWS account. For more information, see Creating an Amazon Chime SDK Voice Connector in the *Amazon Chime SDK Administrator Guide*.
2. Edit your Amazon Chime SDK Voice Connector settings to allow calling from your voice infrastructure to AWS. For more information, see Editing Amazon Chime SDK Voice Connector settings in the *Amazon Chime SDK Administrator Guide*.
   a. For **Termination settings**, select **Enabled**.
   b. For **Allowlist**, choose **New**.
   c. Enter the CIDR notations of the IP addresses for your internal SIP infrastructure. This allows your infrastructure to access the Amazon Chime SDK Voice Connector. For example, to allow traffic from IP address `10.24.34.0`, allowlist the CIDR notation `10.24.34.0/32`.
   d. Choose **Add**.
   e. For **Calling plan**, select the country or countries to add to your calling plan.
   f. Edit any other settings as needed, and choose **Save**.

3. In the Amazon Chime SDK console, under **Voice connectors**, view the **Outbound host name** for your Amazon Chime SDK Voice Connector. For example, *abcdef1ghij2klmno3pqr4*.voiceconnector.chime.aws.
4. To join a meeting using the Amazon Chime SDK, use a SIP URI to make a SIP request to the **Outbound host name** of your Amazon Chime SDK Voice Connector. Use phone number **+17035550122** in the SIP URI. Set the `transport` parameter to use the TLS protocol. Finally, use the unique join token generated by calling the CreateAttendee API action. For more information, see the following example.

Example Example: SIP request

The following example shows the contents of a SIP URI used to make a SIP request to an Amazon Chime SDK Voice Connector.

```
sip:+17035550122@abcdef1ghij2klmno3pqr4.voiceconnector.chime.aws;transport=tls;X-chime-
join-token=join-token
```

The following example shows a sample SIP INVITE message to join an Amazon Chime SDK meeting.

```
INVITE sip:
+17035550122@abcdef1ghij2klmno3pqr4.voiceconnector.chime.aws;transport=tls;X-chime-
join-token=join-token SIP/2.0
Via: SIP/2.0/TLS IPaddress:12345;rport;branch=branch;alias
Max-Forwards: 70
From: sip:+12065550100@IPaddress;tag=tag
To: sip:+17035550122@abcdef1ghij2klmno3pqr4.voiceconnector.chime.aws;X-chime-join-
token=join-token
Contact: <sip:+12065550100@IPaddress:54321;transport=TLS;ob>
Call-ID: a1234567-89b0-1c2d-e34f-5gh678j9k2lm
CSeq: 6214 INVITE
Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, INFO, SUBSCRIBE, NOTIFY, REFER,
 MESSAGE, OPTIONS
Supported: replaces, 100rel, timer, norefersub
Session-Expires: 1800
Min-SE: 90
Content-Type: application/sdp
Content-Length:   991

v=0
o=- 3775321410 3775321410 IN IP4 IPaddress
s=pjmedia
```

```
b=AS:117
t=0 0
a=X-nat:0
m=audio 4000 RTP/SAVP 0 3 8 9 125 101
c=IN IP4 IPaddress
b=TIAS:96000
a=rtcp:4001 IN IP4 IPaddress
a=sendrecv
a=rtpmap:0 PCMU/8000
a=rtpmap:3 GSM/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:9 G722/8000
a=rtpmap:125 opus/48000/2
a=fmtp:125 useinbandfec=1
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=crypto:1 AEAD_AES_256_GCM inline:EXAMPLE
a=crypto:2 AEAD_AES_256_GCM_8 inline:EXAMPLE
a=crypto:3 AES_256_CM_HMAC_SHA1_80 inline:EXAMPLE
a=crypto:4 AES_256_CM_HMAC_SHA1_32 inline:EXAMPLE
a=crypto:5 AES_CM_128_HMAC_SHA1_80 inline:EXAMPLE
a=crypto:6 AES_CM_128_HMAC_SHA1_32 inline:EXAMPLE
```

Note

The Amazon Chime SDK recognizes phone numbers only in E.164 format. Make sure that an E.164 phone number is in your `From` header.

Understanding Amazon Chime SDK event notifications

The Amazon Chime SDK supports sending meeting event notifications to Amazon EventBridge, Amazon Simple Queue Service (SQS), and Amazon Simple Notification Service (SNS).

Note

The default Amazon Chime SDK meetings namespace uses the `ChimeSDKMeetings` endpoints. The legacy Chime namespace uses a single endpoint. For more information about the namespaces and endpoints, refer to Migrating to the Amazon Chime SDK meetings namespace, earlier in this guide.

Sending notifications to EventBridge

You can send Amazon Chime SDK Event notifications to EventBridge. For detailed information about using the Amazon Chime SDK with EventBridge, see Automating the Amazon Chime SDK with EventBridge in the *Amazon Chime SDK Administrator Guide*. For information about EventBridge, see the Amazon EventBridge User Guide.

Sending notifications to Amazon SQS and Amazon SNS

You can use the CreateMeeting API in the *Amazon Chime SDK API Reference* to send Amazon Chime SDK meeting event notifications to one Amazon SQS queue and one Amazon SNS topic per meeting. This can help reduce notification latency. For more information about Amazon SQS, see the Amazon Simple Queue Service Developer Guide. For more information about Amazon SNS, see the Amazon Simple Notification Service Developer Guide.

The notifications sent to Amazon SQS and Amazon SNS contain the same information as the notifications that the Amazon Chime SDK sends to EventBridge. The Amazon Chime SDK supports sending meeting event notifications to queues and topics in the API Region used to create a meeting. Event notifications might be delivered out of order of occurrence.

Granting the Amazon Chime SDK access to Amazon SQS and Amazon SNS

Before the Amazon Chime SDK can send you notifications via an Amazon SQS queue or Amazon SNS topic, you must grant the Amazon Chime SDK permission to publish messages to the Amazon Resource Name (ARN) of the queue or topic. To do this, attach an AWS Identity and Access Management (IAM) policy to the queue or topic that grants the appropriate permissions to the Amazon Chime SDK. For more information, see Identity and access management in Amazon SQS in the *Amazon Simple Queue Service Developer Guide* and Example cases for Amazon SNS access control in the *Amazon Simple Notification Service Developer Guide*.

> **Note**
>
> Your Amazon SQS queue or Amazon SNS topic must use the same AWS region as your Amazon Chime SDK API endpoint.

Example Allow the Amazon Chime SDK to publish events to an Amazon SQS queue

The following example IAM policy grants the Amazon Chime SDK permission to publish meeting event notifications to the specified Amazon SQS queue. Note the conditional statement for `aws:SourceArn` and `aws:SourceAccount`. They address potential Confused Deputy issues.

> **Note**
>
> - You can use `aws:SourceArn` or `aws:SourceAccount` when creating the policies below. You don't need to use both.
> - These examples use the `ChimeSDKMeetings` namespace and corresponding endpoint. If you use the `Chime` namespace, you must use the `chime.amazonaws.com` endpoint.

```
{
    "Version": "2008-10-17",
   "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "meetings.chime.amazonaws.com"
            },
                "Action": [
                    "sqs:SendMessage",
                    "sqs:GetQueueUrl"
                ],
               "Resource": "arn:aws:sqs:"eu-central-1:111122223333:queueName",
               "Condition": {
                   "ArnLike": {
                       "aws:SourceArn": "arn:partition:chime::111122223333:*"
               },
               "StringEquals": {
                   "aws:SourceAccount": "111122223333"
               }
            }
        }
    ]
}
```

This example shows an Amazon SNS policy that allows the Amazon Chime SDK to send meeting event notifications to your SNS topic.

```
{
    "Version": "2008-10-17",
    "Id": "example-ID",
    "Statement": [
     {
        "Sid": "allow-chime-sdk-access-statement-id",
        "Effect": "Allow",
        "Principal": {
            "Service": "meetings.chime.amazonaws.com"
    },
       "Action": [
           "SNS:Publish"
       ],
           "Resource": "arn:aws:sns:eu-central-1:111122223333:topicName",
           "Condition": {
           "ArnLike": {
               "aws:SourceArn": "arn:partition:chime::111122223333:*"
      },
      "StringEquals": {
          "aws:SourceAccount": "111122223333"
          }
       }
     }
   ]
}
```

If the Amazon SQS queue is enabled for server-side encryption (SSE), you must take an additional step. Attach an IAM policy to the associated AWS KMS key that grants the Amazon Chime SDK permission to the AWS KMS actions needed to encrypt data added to the queue.

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "meetings.chime.amazonaws.com"
            },
```

```
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Allow the Amazon Chime SDK to publish events to an Amazon SNS topic

The following example IAM policy grants the Amazon Chime SDK permission to publish meeting event notifications to the specified Amazon SNS topic.

```
{
    "Version": "2008-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "allow-chime-sdk-access-statement-id",
            "Effect": "Allow",
            "Principal": {
                "Service": "meetings.chime.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "arn:aws:sns:eu-central-1:111122223333:topicName",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:partition:chime::111122223333:*"
            },
           "StringEquals": {
               "aws:SourceAccount": "111122223333"
           }
        }
     }
  ]
}
```

Migrating from the Amazon Chime namespace

The Amazon Chime SDK exposes APIs on a set of endpoints. Although you can make HTTPS requests directly to the endpoints, many customers use the AWS SDK in their applications to call the service APIs. The AWS SDK is available in different languages, and it simplifies API calling by encapsulating request signing and retry logic. The AWS SDK includes a namespace for each service endpoint.

When first launched, the Amazon Chime SDK shared a single endpoint with the Amazon Chime application. As a result, solutions used the `Chime` namespace in the AWS SDK to call the Amazon Chime application and the Amazon Chime SDK APIs.

The Amazon Chime SDK now provides dedicated endpoints for each sub-service, such as meetings and PSTN Audio. Each endpoint is addressable through a dedicated namespace in the AWS SDK.

The following topics list the services, namespaces, and endpoints, and describe how to use them in code and with the AWS CLI.

Topics

Endpoints, namespaces, and CLI commands

The following table lists the dedicated Amazon Chime SDK namespaces, endpoints, and CLI commands. The links take you to more information about each service.

Endpoint	AWS SDK Namespace	AWS SDK CLI
identity-chime	ChimeSDKIdentity	chime-sdk-identity
media-pipelines-chime	ChimeSDKMediaPipelines	chime-sdk-media-pipelines
meetings-chime	ChimeSDKMeetings	chime-sdk-meetings
messaging-chime	ChimeSDKMessaging	chime-sdk-messaging

Endpoint	AWS SDK Namespace	AWS SDK CLI
voice-chime	ChimeSDKVoice	chime-sdk-voice

Migration help for each service

All customers should consider using the dedicated Amazon Chime SDK endpoints for access to the latest Amazon Chime SDK features, APIs, and AWS Regions. If you use the shared endpoint with the `Chime` namespace, the following migration guides can help understand the technical differences before migrating.

- Migrating to the Amazon Chime SDKIdentity namespace
- Migrating to the Amazon Chime SDKMediaPipelines namespace
- Migrating to the Amazon Chime SDKMeetings namespace
- Migrating to the Amazon Chime SDKMessaging namespace
- Migrating to the Amazon Chime SDKVoice namespace

API mapping

The following table lists the APIs in the `Chime` namespace, and their corresponding dedicated namespaces and APIs. Some of the dedicated APIs differ from the `Chime` APIs, and the table indicates those instances.

Chime namespace API	Dedicated namespace	Dedicated namespace API
AssociatePhoneNumbersWithVoiceConnector	voice-chime	AssociatePhoneNumbersWithVoiceConnector
AssociatePhoneNumbersWithVoiceConnectorGroup	voice-chime	AssociatePhoneNumbersWithVoiceConnectorGroup
BatchCreateAttendee	meetings-chime	BatchCreateAttendee
BatchCreateChannelMembership	messaging-chime	BatchCreateChannelMembership

Chime namespace API	Dedicated namespace	Dedicated namespace API
CreateAppInstance	identity-chime	CreateAppInstance
CreateAppInstanceAdmin	identity-chime	CreateAppInstanceAdmin
CreateAppInstanceUser	identity-chime	CreateAppInstanceUser
CreateAttendee	meetings-chime	CreateAttendee
CreateChannel	messaging-chime	CreateChannel
CreateChannelBan	messaging-chime	CreateChannelBan
CreateChannelMembership	messaging-chime	CreateChannelMembership
CreateChannelModerator	messaging-chime	CreateChannelModerator
CreateMediaCapturePipeline	media-pipelines-chime	CreateMediaCapturePipeline
CreateMeeting	meetings-chime	CreateMeeting
CreateMeetingWithAttendees	meetings-chime	CreateMeetingWithAttendees
CreateMeetingDialOut*	n/a	
CreateProxySession	voice-chime	CreateProxySession
CreateSipMediaApplication	voice-chime	CreateSipMediaApplication
CreateSipMediaApplicationCall	voice-chime	CreateSipMediaApplicationCall
CreateSipRule	voice-chime	CreateSipRule
CreateVoiceConnector	voice-chime	CreateVoiceConnector
CreateVoiceConnectorGroup	voice-chime	CreateVoiceConnectorGroup
DeleteAppInstance	identity-chime	DeleteAppInstance
DeleteAppInstanceAdmin	identity-chime	DeleteAppInstanceAdmin

Chime namespace API	Dedicated namespace	Dedicated namespace API
DeleteAppInstanceStreamingConfigurations	messaging-chime	DeleteAppInstanceStreamingConfigurations
DeleteAppInstanceUser	identity-chime	DeleteAppInstanceUser
DeleteAttendee	meetings-chime	DeleteAttendee
DeleteChannel	messaging-chime	DeleteChannel
DeleteChannelBan	messaging-chime	DeleteChannelBan
DeleteChannelMembership	messaging-chime	DeleteChannelMembership
DeleteChannelMessage	messaging-chime	DeleteChannelMessage
DeleteChannelModerator	messaging-chime	DeleteChannelModerator
DeleteMediaCapturePipeline	media-pipelines-chime	DeleteMediaCapturePipeline
DeleteMeeting	meetings-chime	DeleteMeeting
DeleteProxySession	voice-chime	DeleteProxySession
DeleteSipMediaApplication	voice-chime	DeleteSipMediaApplication
DeleteSipRule	voice-chime	DeleteSipRule
DeleteVoiceConnector	voice-chime	DeleteVoiceConnector
DeleteVoiceConnectorEmergencyCallingConfiguration	voice-chime	DeleteVoiceConnectorEmergencyCallingConfiguration
DeleteVoiceConnectorGroup	voice-chime	DeleteVoiceConnectorGroup
DeleteVoiceConnectorOrigination	voice-chime	DeleteVoiceConnectorOrigination
DeleteVoiceConnectorProxy	voice-chime	DeleteVoiceConnectorProxy

Chime namespace API	Dedicated namespace	Dedicated namespace API
DeleteVoiceConnectorStreamingConfiguration	voice-chime	DeleteVoiceConnectorStreamingConfiguration
DeleteVoiceConnectorTermination	voice-chime	DeleteVoiceConnectorTermination
DeleteVoiceConnectorTerminationCredentials	voice-chime	DeleteVoiceConnectorTerminationCredentials
DescribeAppInstance	identity-chime	DescribeAppInstance
DescribeAppInstanceAdmin	identity-chime	DescribeAppInstanceAdmin
DescribeAppInstanceUser	identity-chime	DescribeAppInstanceUser
DescribeChannel	messaging-chime	DescribeChannel
DescribeChannelBan	messaging-chime	DescribeChannelBan
DescribeChannelMembership	messaging-chime	DescribeChannelMembership
DescribeChannelMembershipForAppInstanceUser	messaging-chime	DescribeChannelMembershipForAppInstanceUser
DescribeChannelModeratedByAppInstanceUser	messaging-chime	DescribeChannelModeratedByAppInstanceUser
DescribeChannelModerator	messaging-chime	DescribeChannelModerator
DisassociatePhoneNumbersFromVoiceConnector	voice-chime	DisassociatePhoneNumbersFromVoiceConnector
DisassociatePhoneNumbersFromVoiceConnectorGroup	voice-chime	DisassociatePhoneNumbersFromVoiceConnectorGroup
GetAppInstanceRetentionSettings	identity-chime	GetAppInstanceRetentionSettings

Chime namespace API	Dedicated namespace	Dedicated namespace API
GetAppInstanceStreamingConfigurations	messaging-chime	GetMessagingStreamingConfigurations
GetAttendee	meetings-chime	GetAttendee
GetChannelMessage	messaging-chime	GetChannelMessage
GetMediaCapturePipeline	media-pipelines-chime	GetMediaCapturePipeline
GetMeeting	meetings-chime	GetMeeting
GetMessagingSessionEndpoint	messaging-chime	GetMessagingSessionEndpoint
GetProxySession	voice-chime	GetProxySession
GetSipMediaApplication	voice-chime	GetSipMediaApplication
GetSipMediaApplicationLoggingConfiguration	voice-chime	GetSipMediaApplicationLoggingConfiguration
GetSipRule	voice-chime	GetSipRule
GetVoiceConnector	voice-chime	GetVoiceConnector
GetVoiceConnectorEmergencyCallingConfiguration	voice-chime	GetVoiceConnectorEmergencyCallingConfiguration
GetVoiceConnectorGroup	voice-chime	GetVoiceConnectorGroup
GetVoiceConnectorLoggingConfiguration	voice-chime	GetVoiceConnectorLoggingConfiguration
GetVoiceConnectorOrigination	voice-chime	GetVoiceConnectorOrigination
GetVoiceConnectorProxy	voice-chime	GetVoiceConnectorProxy

Chime namespace API	Dedicated namespace	Dedicated namespace API
GetVoiceConnectorStreamingConfiguration	voice-chime	GetVoiceConnectorStreamingConfiguration
GetVoiceConnectorTermination	voice-chime	GetVoiceConnectorTermination
GetVoiceConnectorTerminationHealth	voice-chime	GetVoiceConnectorTerminationHealth
ListAppInstanceAdmins	identity-chime	ListAppInstanceAdmins
ListAppInstances	identity-chime	ListAppInstances
ListAppInstanceUsers	identity-chime	ListAppInstanceUsers
ListAttendees	meetings-chime	ListAttendees
ListAttendeeTags*	n/a	
ListChannelBans	messaging-chime	ListChannelBans
ListChannelMemberships	messaging-chime	ListChannelMemberships
ListChannelMembershipsForAppInstanceUser	messaging-chime	ListChannelMembershipsForAppInstanceUser
ListChannelMessages	messaging-chime	ListChannelMessages
ListChannelModerators	messaging-chime	ListChannelModerators
ListChannels	messaging-chime	ListChannels
ListChannelsModeratedByAppInstanceUser	messaging-chime	ListChannelsModeratedByAppInstanceUser
ListMediaCapturePipelines	media-pipelines-chime	ListMediaCapturePipelines
ListMeetings*	n/a	

Chime namespace API	Dedicated namespace	Dedicated namespace API
ListMeetingTags+	meetings-chime	ListTagsForResource
ListProxySessions	voice-chime	ListProxySessions
ListSipMediaApplications	voice-chime	ListSipMediaApplications
ListSipRules	voice-chime	ListSipRules
ListTagsForResource	identity-chime	ListTagsForResource
ListVoiceConnectorGroups	voice-chime	ListVoiceConnectorGroups
ListVoiceConnectors	voice-chime	ListVoiceConnectors
ListVoiceConnectorTerminationCredentials	voice-chime	ListVoiceConnectorTerminationCredentials
PutAppInstanceRetentionSettings	identity-chime	PutAppInstanceRetentionSettings
PutAppInstanceStreamingConfigurations	messaging-chime	PutMessagingStreamingConfigurations
PutSipMediaApplicationLoggingConfiguration	voice-chime	PutSipMediaApplicationLoggingConfiguration
PutVoiceConnectorEmergencyCallingConfiguration	voice-chime	PutVoiceConnectorEmergencyCallingConfiguration
PutVoiceConnectorLoggingConfiguration	voice-chime	PutVoiceConnectorLoggingConfiguration
PutVoiceConnectorOrigination	voice-chime	PutVoiceConnectorOrigination
PutVoiceConnectorProxy	voice-chime	PutVoiceConnectorProxy

Chime namespace API	Dedicated namespace	Dedicated namespace API
PutVoiceConnectorStreamingConfiguration	voice-chime	PutVoiceConnectorStreamingConfiguration
PutVoiceConnectorTermination	voice-chime	PutVoiceConnectorTermination
PutVoiceConnectorTerminationCredentials	voice-chime	PutVoiceConnectorTerminationCredentials
RedactChannelMessage	messaging-chime	RedactChannelMessage
SendChannelMessage	messaging-chime	SendChannelMessage
StartMeetingTranscription	meetings-chime	StartMeetingTranscription
StopMeetingTranscription	meetings-chime	StopMeetingTranscription
TagAttendee*	n/a	
TagMeeting+	meetings-chime	TagResource
TagResource	identity-chime	TagResource
	media-pipelines-chime	TagResource
	meetings-chime	TagResource
	messaging-chime	TagResource
	voice-chime	TagResource
UntagAttendee*	n/a	
UntagMeeting+	meetings-chime	UntagResource
UntagResource	identity-chime	UntagResource
	media-pipelines-chime	UntagResource
	meetings-chime	UntagResource

Chime namespace API	Dedicated namespace	Dedicated namespace API
	messaging-chime	UntagResource
	voice-chime	UntagResource
UpdateAppInstance	identity-chime	UpdateAppInstance
UpdateAppInstanceUser	identity-chime	UpdateAppInstanceUser
UpdateChannel	messaging-chime	UpdateChannel
UpdateChannelMessage	messaging-chime	UpdateChannelMessage
UpdateChannelReadMarker	messaging-chime	UpdateChannelReadMarker
UpdateProxySession	voice-chime	UpdateProxySession
UpdateSipMediaApplication	voice-chime	UpdateSipMediaApplication
UpdateSipMediaApplicationCall	voice-chime	UpdateSipMediaApplicationCall
UpdateSipRule	voice-chime	UpdateSipRule
UpdateVoiceConnector	voice-chime	UpdateVoiceConnector
UpdateVoiceConnectorGroup	voice-chime	UpdateVoiceConnectorGroup
ValidateE911Address	voice-chime	ValidateE911Address

+ API has been superseded by an API with another name.

***** API is no longer available.

Using Amazon Chime SDK meetings

The topics in this section explain how to use Amazon Chime SDK meetings to create custom meeting applications. We recommend following these topics in the order listed.

Topics

Migrating to the Amazon Chime SDK meetings namespace

The Amazon Chime SDK Meetings namespace is a dedicated place for the APIs that create and manage Amazon Chime SDK meeting resources. You use the namespace to address Amazon Chime SDK meeting API endpoints in any AWS Region in which they're available. Use this namespace if you're just starting to use the Amazon Chime SDK. For more information about Regions, refer to Available AWS Regions for the Amazon Chime SDK service in this guide.

Existing applications that use the Amazon Chime namespace should plan to migrate to the dedicated namespace in order to use the latest APIs and features.

Topics

- Differences between the namespaces

Reasons to migrate

We encourage you to migrate to the Amazon Chime SDK Meetings namespace for these reasons:

Choice of API Endpoint

The Amazon Chime SDK Meetings namespace is the only API namespace which can use API endpoints in any region that makes them available. If you want to use API endpoints other than `us-east-1`, you must use the Amazon Chime SDK Meetings namespace.

For more information about how Amazon Chime SDK meetings use AWS Regions, refer to Meeting Regions in this guide.

Updated and new meeting APIs

We only add or update meeting APIs in the Amazon Chime SDK Meetings namespace.

Before you migrate

Before you migrate, be aware of the differences between the namespaces. The following table lists and describes them.

	Amazon Chime SDK Meetings namespace	Amazon Chime namespace
AWS SDK namespace	ChimeSDKMeetings	Chime
Regions	Multiple	us-east-1 only
Endpoints	https://meetings-chime.*region*.amazonaws.com	https://service.chime.aws.amazon.com
Service principal	meetings.chime.amazonaws.com	chime.amazonaws.com
APIs	Only APIs for meetings	APIs for meetings and other parts of Amazon Chime

	Amazon Chime SDK Meetings namespace	Amazon Chime namespace
CreateMeeting	`ExternalMeetingId` and `MediaRegion` are required.	`ExternalMeetingId` and `MediaRegion` are optional.
CreateMeetingWithAttendees	`ExternalMeetingId` and `MediaRegion` are required.	`ExternalMeetingId` and `MediaRegion` are optional.
`ListMeetings`	Not available	Available
`ExternalMeetingId`	Validation includes pattern matching	Available
`ExternalUserId`	Validation includes pattern matching	Available
Meeting Tags APIs	`TagResource` , `UntagResource` , `ListTagsForResource`	`TagMeeting` , `UntagMeeting` , `ListMeetingTags`
Attendee Tags	Not available	Available
Echo reduction	Available	Not available
Live transcription language identification	Available	Not available
Attendee capabilities	Available	Not available
Media replication	Available	Not available
AppKeys and TenantIds	Available	Not available

	Amazon Chime SDK Meetings namespace	Amazon Chime namespace
Media pipelines	Media pipelines support multiple regions in the Amazon Chime SDK Meetings namespace. For more information, see Migrating to the ChimeSdkMediaPipelines namespace.	Available via the us-east-1 endpoint
SIP media application	`JoinChimeMeeting` action requires `MeetingId`	`JoinChimeMeeting` action does not require `MeetingId`
Direct SIP integration	Not available	Available

Differences between the namespaces

The following sections explain the differences between the `Amazon Chime` and `Amazon Chime SDK Meetings` namespaces.

AWS SDK namespace

The Amazon Chime SDK namespace uses the `Chime` formal name. The Amazon Chime SDK Meetings namespace uses the `ChimeSDKMeetings` formal name. The precise format of the name varies by platform.

For example, if you use the AWS SDK in Node.js to create meetings, you use a line of code to address the namespace.

```
const chimeMeetings = AWS.Chime();
```

To migrate to the Amazon Chime Meetings SDK, update this line of code with the new namespace and the endpoint region.

```
const chimeMeetings = AWS.ChimeSDKMeetings({ region: "eu-central-1" });
```

Regions

The Amazon Chime namespace can only address API endpoints in the us-east-1 Region. The Amazon Chime SDK Meetings namespace can address Amazon Chime SDK meeting API endpoints in any Region they are available. For a current list of meeting Regions, refer to Available AWS Regions for the Amazon Chime SDK service in this guide.

Endpoints

The Amazon Chime SDK Meetings namespace uses different API endpoints than the Amazon Chime namespace.

Only the endpoint used to create a meeting can be used to modify it. This means a meeting created via an endpoint in EU-CENTRAL-1 can only be modified via EU-CENTRAL-1. It also means you cannot address a meeting created via the `Chime` namespace with the `ChimeSDKMeetings` namespace in `US-EAST-1`. For more information about the current endpoints, refer to API mapping in this guide.

Service principal

The Amazon Chime SDK Meetings namespace uses a new service principal: `meetings.chime.amazonaws.com`. If you have SQS, SNS, or other IAM access policies that grant access to the service, you need to update those polices to grant access to the new service principal.

APIs

The Amazon Chime SDK Meetings namespace only contains APIs for creating and managing meetings. The Amazon Chime namespace includes APIs for meetings and other parts of the Amazon Chime service.

CreateMeeting required fields

In the Amazon Chime SDK Meetings namespace, the CreateMeeting and CreateMeetingWithAttendees APIs require the `ExternalMeetingId` and `MediaRegion` fields to be specified.

External ID values

The Amazon Chime SDK Meetings namespace enforces addition validation on the values that can be used for `ExternalMeetingId` and `ExternalUserId`.

Echo reduction

Amazon Chime SDK Meetings namespace offers machine learning-based echo reduction to help remove noise and sound from the local loudspeaker from circulating back into the meeting. Refer to the guide on GitHub for more information.

Attendee capabilities

Amazon Chime SDK Meetings namespace provides granular control over an attendees capabilities within a meeting to send and receive audio, video and content.

Media replication

Amazon Chime SDK Meetings namespace offers media replication to link a primary meeting to replica meetings to bring together up to 10,000 people for a real-time session. Participants connected to a replica session receive the media of the presenters connected to the primary session, but they can be promoted to the primary meeting. For more information, refer to Using media replication in this guide.

AppKeys and TenantIds

Amazon Chime SDK Meetings namespace provides a way to limit access from a network to specific Amazon Chime SDK meetings. For more information, refer to Using AppKeys and TenantIDs in this guide.

Media pipelines

Amazon Chime SDK media pipelines work with meetings created by any meetings endpoint, with either the Amazon Chime SDK Meetings or the Amazon Chime namespace. Refer to Available regions for the latest list of media pipeline regions.

SIP media applications

Amazon Chime SDK SIP media applications work with meetings created by any meetings endpoint, with either the Amazon Chime SDK Meetings or the Amazon Chime namespace. When using SIP media applications with a meeting created through the Amazon Chime SDK Meetings namespace, the JoinChimeMeeting action requires the `MeetingId` parameter.

Additional APIs

The Meetings namespace has a growing list of APIs that the Chime namespace does not have. If you are getting started with the Amazon Chime SDK, use the Meetings namespace to access the latest features.

Using meeting Regions

Amazon Chime SDK meetings have *control* Regions and *media* Regions. Control Regions have an API endpoint used to create, update, and delete meetings. Media Regions host the actual meetings.

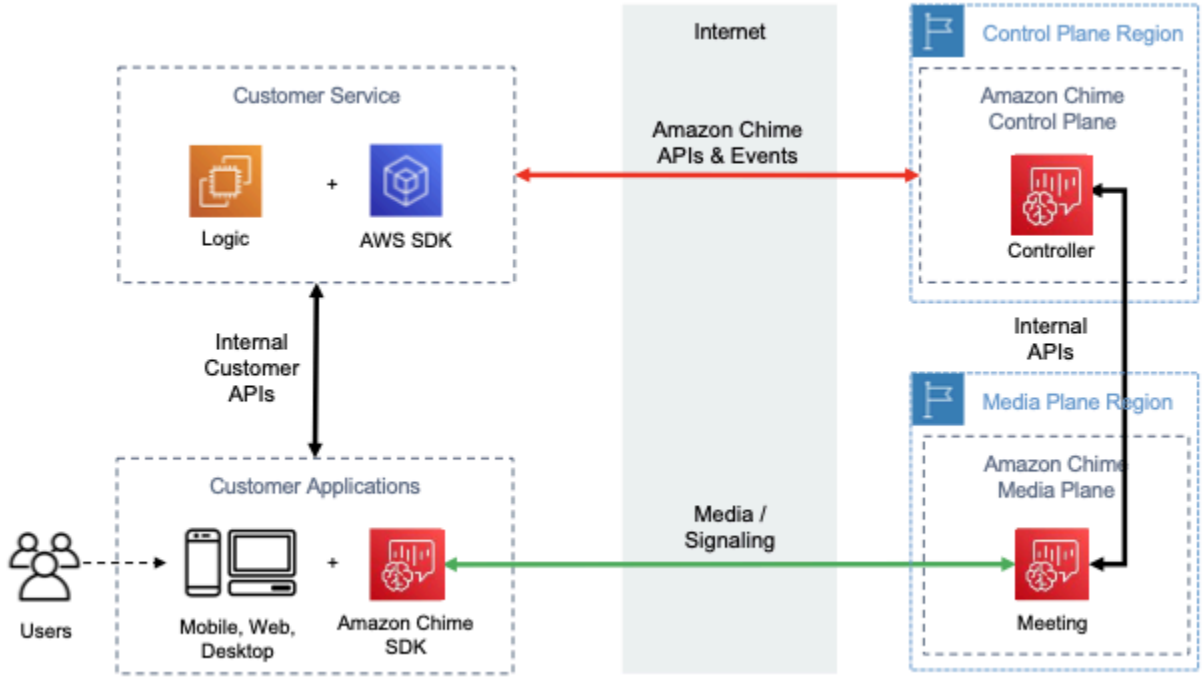
Typically, your application service uses the AWS SDK to sign and call APIs in control Regions. Your application client uses the Amazon Chime SDK client libraries for JavaScript, iOS, or Android to connect to the meeting in media Regions.

A control region can create a meeting in any media Region in the same AWS partition. However, you can only update a meeting in the control region used to create it. To find the media Region closest to a customer, call https://nearest-media-region.l.chime.aws.

Meeting events such as `AttendeeJoined` call EventBridge, Amazon Simple Queue Service (SQS), or Amazon Simple Notification Service (SNS) in the meeting control Region.

For a list of available Amazon Chime SDK meeting control and media Regions, refer to Available AWS Regions for the Amazon Chime SDK service in this guide.

This diagram shows the typical flow of data through the control and media Regions.



Choosing a control region

Remember these factors when choosing a control Region for an Amazon Chime SDK meeting:

- **Regulatory requirements**. Is your application required to be within a geopolitical border, or use an endpoint with FIPS 140-2 validated cryptographic modules?
- **API latency**. Using the control Region nearest to the AWS Region of your application service can help reduce the APIs' network latency. In turn, that helps reduce the time needed to create meetings, and let users join meetings faster.
- **High Availability**. You can use multiple control Regions to implement a high availability architectures. However each control Region operates independently. Also, you can only update meetings in the control Region used to create them. Further, you must use that same region to consume meeting events with EventBridge, Amazon Simple Queue Service (SQS), or Amazon Simple Notification Service (SNS).

Choosing a media region

> **Note**
>
> We recommend that you always specify a value in the `MediaRegion` parameter in the CreateMeeting API action. For more information about the Regions, refer to Available AWS Regions for the Amazon Chime SDK service.

When choosing a media Region to use for your Amazon Chime SDK meeting, consider these common factors:

Regulatory requirements

If your Amazon Chime SDK meetings are subject to regulations requiring them to be hosted within a geopolitical border, consider hard coding the meeting Region based on fixed application logic.

For example, a telemedicine application might require all meetings to be hosted within the medical practitioner's jurisdiction. If the application supports clinics located in both Europe and the United States, you can use each clinic's address to select a Region within its jurisdiction.

Meeting quality

When an Amazon Chime SDK meeting is hosted in a media Region, each attendee's audio and video is sent and received from that Region. As the distance between the attendee and the Region increases, meeting quality can be affected by network latency. Specifying a Region for your Amazon Chime SDK meeting can help enhance the meeting quality for your attendees, whether they are located near each other or distributed geographically.

You can use one of the following methods to choose a media Region for your Amazon Chime SDK meeting:

Hard code a media Region

Recommended if your Amazon Chime SDK meetings are all hosted within a specific AWS Region.

Choose the nearest media Region

Recommended if your Amazon Chime SDK meeting attendees are located in the same AWS Region, but your meetings are hosted in different Regions.

Finding the nearest media Region

To find the nearest media Region capable of hosting an Amazon Chime SDK meeting, call https://nearest-media-region.l.chime.aws. This endpoint returns a single Region, such as `{"region": "us-west-2"}`. Call the URL from your client application to identify the Region closest to the user, and then use the result in the `MediaRegion` parameter of the CreateMeeting API to create the meeting in that Region.

You usually call the URL when the client application starts, or its network connection changes. By predetermining the nearest Region, you avoid adding the call's latency at the time of meeting creation.

Finding the nearest AWS GovCloud (US) media Region

To find the nearest AWS GovCloud (US) Region that can host an Amazon Chime SDK meeting, call https://nearest-us-gov-media-region.l.chime.aws. This endpoint returns the nearest region, such as `{"region": "us-gov-west-1"}`. Call the URL from your client application to identify the

AWS GovCloud (US) closest to the user, and use the result in the `MediaRegion` parameter of the CreateMeeting API to create the meeting in that Region.

You usually call the URL when the client application starts, or its network connection changes. By predetermining the nearest Region, you avoid adding the call's latency at the time of meeting creation.

JavaScript example

The following example uses HTML and JavaScript to return the nearest media Region and AWS GovCloud (US) media Region.

```
<html>
<head>
  <title>Amazon Chime SDK - Nearest Media Region</title>
  <script>

async function getNearestMediaRegion(partition)  {

    console.log('Nearest media region partition: ' + partition);

    const url = ('aws-us-gov' == partition) ? 'https://nearest-us-gov-media-
region.l.chime.aws' : 'https://nearest-media-region.l.chime.aws';
    let result = ('aws-us-gov' == partition) ? 'us-gov-west-1' : 'us-west-2';

    try { //Find the nearest media region
        console.log('Nearest media region URL: ' + url);
        const response = await fetch(url, {method: 'GET'} );
        const body = await response.json();
        result = body.region;
    } catch (error) {
        console.log(error.message);
    } finally {
        console.log('Nearest media region found: ' + result);
        return result;
    }
}

async function findRegions(partition) {
  aws.innerText = await getNearestMediaRegion();
  awsusgov.innerText = await getNearestMediaRegion('aws-us-gov');
}
  </script>
```

```
</head>
<body>
  <h3>Nearest media region, by AWS partition</h3>
  <table>
    <tr><th>Partition</th><th>Media Region</th></tr>
    <tr><td>aws</td><td id="aws">Finding...</td></tr>
    <tr><td>aws-us-gov</td><td id="awsusgov">Finding...</td></tr>
  </table>
  <script>
    findRegions();
  </script>
</body>
</html>
```

Checking Region status

Call https://region.status.chime.aws/ to retrieve the health of the Amazon Chime SDK service in each Region. The result shows the recommended Regions. If a media Region has a status other than **recommended**, the nearest media Region endpoint will not return that Region.

The following example shows a typical result.

```
{
  "MeetingsControlRegions": {
    "us-east-1": "recommended",
    "us-west-2": "recommended",
    "ap-southeast-1": "recommended",
    "eu-central-1": "recommended"
  },
  "MeetingsMediaRegions": {
    "af-south-1": "recommended",
    "ap-northeast-1": "recommended",
    "ap-northeast-2": "recommended",
    "ap-south-1": "recommended",
    "ap-southeast-1": "recommended",
    "ap-southeast-2": "recommended",
    "ca-central-1": "recommended",
    "eu-central-1": "recommended",
    "eu-north-1": "recommended",
    "eu-south-1": "recommended",
    "eu-west-1": "recommended",
    "eu-west-2": "recommended",
    "eu-west-3": "recommended",
```

```
    "sa-east-1": "recommended",
    "us-east-1": "recommended",
    "us-east-2": "recommended",
    "us-west-1": "recommended",
    "us-west-2": "recommended"
  },
  "MediaPipelineControlRegions": {
    "ap-southeast-1": "recommended",
    "eu-central-1": "recommended",
    "us-east-1": "recommended",
    "us-west-2": "recommended"
  },
  "MediaPipelineDataRegions": {
    "af-south-1": "recommended",
    "ap-northeast-1": "recommended",
    "ap-northeast-2": "recommended",
    "ap-south-1": "recommended",
    "ap-southeast-1": "recommended",
    "ap-southeast-2": "recommended",
    "ca-central-1": "recommended",
    "eu-central-1": "recommended",
    "eu-north-1": "recommended",
    "eu-south-1": "recommended",
    "eu-west-1": "recommended",
    "eu-west-2": "recommended",
    "eu-west-3": "recommended",
    "sa-east-1": "recommended",
    "us-east-1": "recommended",
    "us-east-2": "recommended",
    "us-west-1": "recommended",
    "us-west-2": "recommended"
  }
}
```

Creating meetings

The following procedure demonstrates how to create a meeting with audio and video for your server and client applications. Before you begin, you must integrate your client application with an Amazon Chime SDK client library. For more information, refer to Learn about the Amazon Chime SDK client libraries.

To create a meeting with audio and video

1. Complete the following steps from your server application:

   a. Use the CreateMeeting API action in the *Amazon Chime SDK API Reference* to create a meeting. Specify an AWS Region using the `MediaRegion` parameter. For more information about choosing a meeting Region, refer to Meeting Regions.

   b. Add attendees to the meeting using the CreateAttendee API action or the BatchCreateAttendee API action. Securely transfer the meeting and attendee from your server application to the client authorized as the respective attendee. For more information about meetings and attendees, refer to Meeting and Attendee in the *Amazon Chime SDK API Reference*.

2. Complete the following steps from your client application:

   a. Use an Amazon Chime SDK client library to construct a `MeetingSessionConfiguration` object. Use the meeting and attendee information from the previous steps.

   b. Implement the `AudioVideoObserver` interface.

   c. Create a `MeetingSession` using the `MeetingSessionConfiguration`.

   d. Use the `AudioVideoFacade` from the `MeetingSession` to control real-time media.

      i. Register an instance of the `AudioVideoObserver` interface. This lets you receive events when the meeting state changes.

      ii. Select initial devices for the audio input, audio output, and video input.

      iii. Start the audiovisual session.

      iv. Start local video capture when the user wants to share video.

      v. To show video tiles, manage video tile events, and bind the tiles to video surfaces in the client application.

      vi. Manage other user interactions such as muting and unmuting, or starting and stopping local video capture.

      vii. To leave the meeting, stop the audiovisual session.

   e. (Optional) Use the `AudioVideoFacade` from the `MeetingSession` to share media content, such as screen captures, with other clients.

      i. Start the screen share session. The content joins the meeting as an additional attendee.

ii. To view the shared content, manage video tile events and bind the tiles to surfaces in the client application.

iii. Manage other interactions, such as pausing, restarting, or stopping the content share.

Meetings end when you run the DeleteMeeting API action. Also, meetings end automatically when:

- The meeting time exceeds 24 hours.
- The meeting is a replica meeting and the primary meeting ends.
- In a non-replica meeting, no attendees connected for five continuous minutes.

Selecting meeting features

When you call the CreateMeeting API, you can specify features to make available to the clients that join the session. Note that some feature options incur additional billing.

The following features are available for sessions:

- `Audio.EchoReduction` – Machine learning echo reduction.
- `Video.MaxResolution` – Maximum webcam video resolution.
- `Content.MaxResolution` – Maximum content sharing resolution..
- `Attendees.MaxCount` – Mmaximum number of attendees.

Topics

Using Audio.EchoReduction

Use `Audio.EchoReduction` to help keep sound from a user's loudspeaker from circulating back into meeting.

Echo reduction is ideal for situations in which a user's loudspeaker will be the primary output device for meeting audio. For example, when multiple users are attending a meeting from the same device in a conference room, or when an individual remote attendee is not wearing headphones.

Echo reduction is available in the JavaScript and React client libraries. For more information, refer to the documentation on GitHub. Additional costs apply, refer to the Amazon Chime SDK Pricing page for details.

Using Video.MaxResolution

Use `Video.MaxResolution` to specify the maximum webcam video resolution for the meeting. The feature provides the following options:

- None: no camera video allowed
- HD: high-definition camera video (1280x720p)
- FHD: full-high-definition camera video (1920x1080)

If FHD (1080p) Video is requested, a high-definition WebRTC session is created. Refer to the Amazon Chime SDK Pricing page for details.

If a client attemps to send webcam video above a specified maximum, the service rejects the video and sends the following error:

```
Disabled video/content send capability, reason: Video resolution is above
limit of current meeting feature selection.
```

Using Content.MaxResolution

Use `Content.MaxResolution` to specify the maximum content share resolution for the meeting. The feature provides the following options:

- None: no content share allowed
- FHD: full-high-definition content share (1920x1080)
- UHD: ultra-high-definition content share (3840x2160)

If UHD (4K) content is requested, a high-definition WebRTC session is created.

If a client attems to send a content share beyond the maximum resolution, that resolution is scaled down to the specified maximum. You scale by applying `MediaTrackConstraints` to the content share track. The following examples shows how to scale a share track.

```
const constraint: MediaTrackConstraints = {
    width: { ideal: videoQualitySettings.videoWidth },
    height: { ideal: videoQualitySettings.videoHeight },
    frameRate: { ideal: videoQualitySettings.videoFrameRate },
  };
  this.context.logger.info(
    `Video track (content = ${isContentAttendee}) with constraint: ${JSON.stringify(
      constraint
    )}, trackSettings: ${JSON.stringify(trackSettings)}`
  );
  try {
    await mediaStreamTrack.applyConstraints(constraint);
  } catch (error) {
    this.context.logger.info(
      `Could not apply constraint for video track (content = ${isContentAttendee})`
    );
  }
```

The following table shows the expected behavior for content sharing.

Content feature	Content share native resolution	Scaling	Content coding resolution
FHD	1280x720	No	1280x720
FHD	1920x1080	No	1920x1080
FHD	3840x2160	Yes	1920x1080
UHD	1920x1080	No	1920x1080
UHD	3840x2160	No	3840x2160
UHD	4200x2400	Yes	3780x2160

Using Attendees.MaxCount

Use `Attendee.MaxCount` to specify the maximum number of attendees allowed into a meeting. The upper limit of Attendee.MaxCount depends on the session type. For a standard session, you can select a maximum of 250 attendees. For a high-definition session, you *must* select a value of up to 25 attendees.

If you request FHD (1080p) video or UHD (4K) content, your session will be a high-definition session.

Attendee capacity costs apply for high-definition sessions. Refer to the Amazon Chime SDK Pricing page for details.

Using meeting features in a client app

Creating a meeting with specified features

To create a meeting, call the CreateMeeting API and specify the desired meeting features. The following example shows how to specify all the features.

```
// You must migrate to the Amazon Chime SDK Meetings namespace.
const chime = AWS.ChimeSDKMeetings({ region: "eu-central-1" });

// Create meeting
const meetingInfo = await chime.createMeeting({
    ...
    MeetingFeatures: {
      Audio: {
        EchoReduction: 'AVAILABLE'
      },
      Video: {
        MaxResolution: 'FHD'
      },
      Content: {
        MaxResolution: 'UHD'
      },
      Attendee: {
        MaxCount: 25
      },
    }
```

```
  }).promise();
```

Using meeting features in a client

After you create a meeting with the desired features, you can pass in the `joinInfo` when you create the `MeetingSessionConfiguration` object. The meeting features are used at `meetingSession` creation to set webcam video resolution and bitrate, and the content share resolution and bitrate.

```
const configuration = new MeetingSessionConfiguration(this.joinInfo.Meeting,
 this.joinInfo.Attendee);

this.meetingSession = new DefaultMeetingSession(
    configuration,
    this.meetingLogger,
    this.deviceController,
    new DefaultEventController(configuration, this.meetingLogger, this.eventReporter)
);
```

How the Amazon Chime SDK uses WebRTC media

The Amazon Chime SDK supports two types of WebRTC sessions, standard and high-definition. The following topics describe the media available in each type of session when using the Amazon Chime SDK client libraries for JavaScript, React, iOS, and Android.

Topics

Audio

Each Amazon Chime client sends one audio stream to the sessions and receives one audio stream from the session. Typically, microphones on local devices generate the audio. The audio received is a mix of the audio sent from the other session clients.

Both session types support sample rates up to 48kHz and up to 2 channels (stereo) encoded with bitrates up to 128kbps using the Opus codec. However, the audio streams sent and received vary by client library type:

- The Amazon Chime SDK client libraries for JavaScript and React support sending and receiving mono and stereo audio at the highest sample rate supported by the device and browser, up to a maximum of 48kHz.
- The Amazon Chime SDK client libraries for iOS and Android support sending mono audio up to 48kHz, and receiving stereo audio at 48kHz.

Video

Each Amazon Chime client can send one video stream to the session and receive up to 25 video streams from the session. The video sent is typically sourced from the local device's webcam. Each client can select up to 25 video streams to receive, and change the selection at any time during the session.

Standard sessions support video resolutions up to 1280x720 at 30 frames per second encoded with bitrates up to 1500kbps using H.264, VP8, VP9, and AV1.

High-definition sessions support video resolutions up to 1920x1080 at 30 frames per second encoded with bitrates up to 2500kbps using H.264, VP8, VP9, and AV1.

The Amazon Chime SDK client libraries for JavaScript and React support sending video in simulcast at 15 frames per second, or with scalable video coding (SVC). SVC encodes a single video stream with three spatial layers and three temporal layers at 100%, 50%, and 25% of your target values. The service automatically selects the layer to send to each viewer based on the viewers' available bandwidth.

The Amazon Chime SDK client libraries for iOS and Android support sending up to 15 frames per second. However, the actual frame rate and resolution is automatically managed by the Amazon Chime SDK.

Video encoding and decoding uses hardware acceleration where available to improve performance.

If a client sends video with a bitrate greater than the maximum allowed bitrate, the session first starts sending the client Receiver Estimated Maximum Bitrate messages via the Real-Time Control Protocol. If the client continues to send video with a bitrate greater than the maximum allowed bitrate, the session discards the incoming video stream packets.

Content share

Up to two clients can share content to the session. A content share can include a video track, an audio track, or both. A common example of a content share is screen share, which uses screen capture as the source of the content. Another example is sharing prerecorded content with video and audio tracks.

Content audio is mixed into the audio stream sent by the session. Content audio supports sample rates up to 48kHz and up to 2 channels (stereo) encoded with bitrates up to 128kbps using the Opus codec.

Video content is sent to the session and forwarded to clients in a separate video stream. Standard sessions support content video up to 1920x1080 at 30 frames per second. High-definition sessions support content video up to 3840 x 2160 at 30 frames per second.

Screen capture for content sharing uses the resolution of the screen or window being captured, up to the maximum content resolution for the session type, and up to 30 frames per second. However, device and browser capabilities may limit those values.

The Amazon Chime SDK client libraries for JavaScript and React support content share from screen capture and other sources.

The Amazon Chime SDK client libraries for iOS and Android only support content share from screen capture.

Data messages

Data messages provide a way for a client to broadcast information to other clients in the session. For example, an application may use data messages to share emoji reactions during a session.

Each data message includes:

- A topic, a string of up to 64 characters.
- Up to 2 KB of data, including the topic.

A client sends a data message to the session, and the session sends the data message to all connected clients.

The session can optionally cache the data message for up to five minutes. If a client joins or reconnects to a session, the session will automatically send the client any cached data messages that have not been previously sent. The session cache stores a maximum of 1024 data messages.

A session supports up to 100 sent data messages per second. When using live transcription, each client receives transcription messages via data messages, which are counted towards the total sent messages per second.

Configuring video codecs

A client device uses a video codec to compress raw video before it is sent to the service, and to decompress received video before it's rendered.

When using the Amazon Chime SDK client library for JavaScript, you can specify your codec preferences for sending video.

The Amazon Chime SDK client libraries for iOS and Android automatically select the codec for you, based on the device's capabilities.

Setting video codec preferences

In the Amazon Chime SDK client library for JavaScript, you can specify independent video codec preferences for webcam and content video.

Use the AudioVideoControllerFacade.setVideoCodecSendPreferences function to set your codec preferences for sending webcam video. The link takes you to GitHub.

Your preferences are passed as an ordered array, with your most preferred codec first and your least preferred codec last.

When providing multiple codec preferences, the service automatically selects the most preferred codec that all session attendees can decode.

The following example shows how to set a video codec preference for VP9 with a fallback option of VP8:

```
// A meeting session has already been created and stored in `this.meetingSession`
this.meetingSession.audioVideo.setVideoCodecSendPreferences(
 [
```

```
  VideoCodecCapability.vp9(),
  VideoCodecCapability.vp8()
 ]
);
```

The following scenarios apply to the preferences:

- **Optimal** – The client encodes video using the VP9 codec.
- **Local Fallback** – If the client doesn't support VP9 encoding, it falls back to VP8 encoding. If the client does not support VP8 encoding, it falls back to any codec supported by the browser and the service.
- **Remote Fallback** – If another client in the session does not have a VP9 decoder, this client falls back to VP8 encoding.
- **Local Failure** – If the client does not support VP9 or VP8 encoding, it will not send video.

To set preferences for content sharing, use the ContentShareControllerFacade.setContentShareVideoCodecPreferences function to set your codec preferences for sending content video. This link takes you to GitHub.

The following example sets a content video codec preference for VP9 with a fallback option of VP8.

```
// A meeting session has already been created and stored in `this.meetingSession`
    this.meetingSession.audioVideo.setContentShareVideoCodecPreferences(
 [
 VideoCodecCapability.vp9(),
 VideoCodecCapability.vp8()
 ]
);
```

Configuring your network for the Amazon Chime SDK

When you integrate the Amazon Chime SDK into your client application, the SDK connects to its back-end service to send and receive audio, video, content sharing, and data messages. If your users' network blocks traffic to the Amazon Chime SDK service, their ability to use the service will be impaired. Network administrators can use this information to reconfigure their network to allow access to the Amazon Chime SDK service.

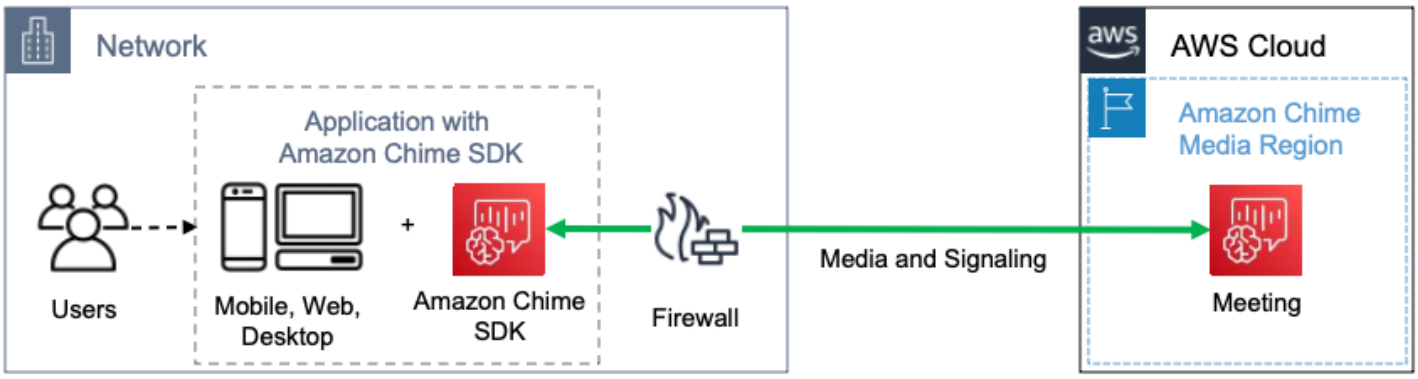
> **Note**
>
> When you configure your network, you must enable Extension Mechanisms for DNS (EDNS0) by default. This enables your application to reach the Amazon Chime SDK services by ensuring that host information is the correct size for UDP packets.

Topics

Configuring for media and signaling

Amazon Chime SDK audio, video, and content use User Datagram Protocol (UDP) transport whenever possible. If UDP is blocked, the Amazon Chime SDK tries to establish a Transport Layer Security (TLS) connection for bidirectional media transport. Amazon Chime SDK signaling and data messages use Transmission Control Protocol (TCP) and WebSocket connections.

The following diagram shows a typical network with an application that runs the Amazon Chime SDK.



The Amazon Chime SDK uses the following destinations and ports for media and signaling.

Domain	Subnet	Ports
*.chime.aws	99.77.128.0/18	TCP:443 UDP:3478

This subnet is the `CHIME_MEETINGS` service in the AWS IP address ranges.

Configuring for Amazon Voice Focus

The Amazon Chime SDK client libraries for iOS and Android include the Amazon Voice Focus module. The Amazon Chime SDK client library for JavaScript downloads the Amazon Voice Focus module from Amazon CloudFront. The Amazon Chime SDK client library for Windows doesn't support Voice Focus.

Amazon Voice Focus uses the following destinations and ports.

Domain	Ports
*.sdkassets.chime.aws	TCP:443

This subnet is the `CLOUDFRONT` service in the AWS IP address ranges.

Configuring for echo reduction

The Amazon Chime SDK client library for JavaScript downloads the echo reduction module from Amazon CloudFront.

Echo reduction uses the following destinations and ports.

Domain	Ports
*.sdkassets.chime.aws	TCP:443

This subnet is the `CLOUDFRONT` service in the AWS IP address ranges.

Configuring for background replacement and blur

The Amazon Chime SDK client library for JavaScript downloads the background replacement and blur module from Amazon CloudFront.

Background replacement and blur uses the following destinations and ports.

Domain	Ports
*.sdkassets.chime.aws	TCP:443

This subnet is the `CLOUDFRONT` service in the AWS IP address ranges.

Configuring browser content security policies

When you build an application with the Amazon Chime SDK client library for JavaScript, you need to configure the browser content security policies in your application. For more information, refer to the Content Security Policy Guide on GitHub.

Using AppKeys and TenantIDs

You can use AppKeys and TenantIDs to limit access *from a network* to specific applications' Amazon Chime SDK WebRTC media sessions.

Developers use the Amazon Chime SDK to create applications that send and receive real-time video over UDP. Application users require UDP access to the `CHIME_MEETINGS` subnet. Organizations (network owners) can use AppKeys and TenantIDs to limit access from their network to only a specific application's WebRTC media sessions.

Example 1: Using AppKeys

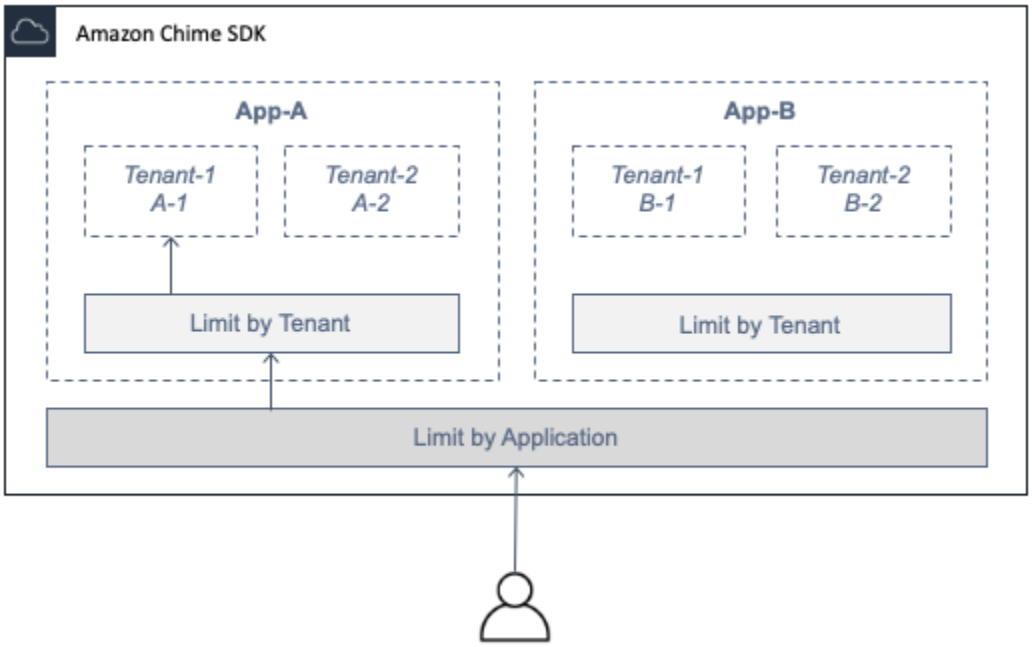
If App-A and App-B use the Amazon Chime SDK, an organization can allow App-A to access the WebRTC media sessions from their network, but block App-B and any other applications that use the Amazon Chime SDK. Organizations can do that with App-A's AppKey and an HTTPS proxy. For more information, refer to Limiting access to a specific application, later in this topic.

Example 2: Using AppKeys and TenantIDs

If App-A is publicly available and used by many customers, an organization may want to allow App-A to access WebRTC media sessions from their network only when their users are part of

the session, and block access to all other App-A sessions. Organizations can do this by using the application's AppKey, the organization's TenantID, and an HTTPS proxy. For more information, refer to Limiting access to a specific tenant, later in this topic.

To use AppKeys and TenantIDs, you must have an HTTPS proxy server that allows adding HTTPS headers to a request. The following diagram shows how AppKeys and TenantIDs work.



In the image, App-A has tenants A-1 and A-2, and App-B has tenants B-1 and B-2. In this case, the AppKey only allows App-A to connect to the WebRTC media session, and the tenant ID only admits Tenant A-1 to the session.

Topics

- Limiting access to a specific application
- Limiting access to a specific tenant
- HTTPS header examples

Limiting access to a specific application

An *AppKey* is a consistent, unique 256-bit value that Amazon Chime creates for each AWS account. If you don't have an AppKey, you can request one from Amazon Support. If you have multiple AWS accounts, you can request a common AppKey for all your accounts.

> **Note**
>
> You can safely share your AppKeys publicly and enable other organizations to limit access from their networks.

The Amazon Chime SDK automatically associates each WebRTC media sessions with an AppKey based on the AWS account ID used to create the session. To limit access *from your network* to specific applications, do the following:

1. Route all outbound requests to the `CHIME_MEETINGS` subnet through an HTTPS proxy server.
2. Configure the proxy server to add the following header to all outbound requests to the `CHIME_MEETINGS` subnet:

   `X-Amzn-Chime-App-Keys:` *`comma-separated list of allowed AppKeys`*.

   For example, `X-Amzn-Chime-App-Keys:`*`AppKey-A`*`,`*`AppKey-B`*`,`*`AppKey-C`* allows the apps associated with those AppKeys to access the subnet.

The Amazon Chime SDK inspects inbound WebRTC media session connections for the `X-Amzn-Chime-App-Keys` header and applies the following logic:

1. If the `X-Amzn-Chime-App-Keys` header is present and includes the session's AppKey, accept the connection.
2. If the `X-Amzn-Chime-App-Keys` header is present but does not include the session's AppKey, reject the connection with a 403 error.
3. If the `X-Amzn-Chime-App-Keys` header is not present, accept the connection. If users can access the application from outside the organization's network, they can also access the session.

Limiting access to a specific tenant

A *TenantID* is an opaque identifier created by developers. Remember the following about TenantIDs:

- TenantIDs are not guaranteed to be unique between applications, so you must specify an AppKey for each TenantID list.
- TenantIDs are case senstitive. Enter them exactly as prescribed by the developer.

- An organization can limit access to multiple applications, but only specify TenantIDs for some of those applications. Applications without TenantIDs can connect to all WebRTC media sessions.

To associate a media session with TenantIDs, a developer must first add the `TenantIds` property and a list of TenantIDs to a CreateMeeting or CreateMeetingWithAttendees request.

For example:

```
CreateMeeting(..., TenantIds : [ tenantId1, tenantId2 ] )
```

To limit access from an organization's network to their WebRTC media session in specific applications, do the following:

1. Follow the steps in Limiting access to a specific application.
2. Configure the HTTPS proxy server to add an `X-Amzn-Chime-Tenants` header on outbound connections. Include a list of AppKeys and TenantIDs, delimited as shown in this example: `X-Amzn-Chime-Tenants: AppKey-A:tenantId-A-1,tenantId-A-2;AppKey-B:tenantId-B-1,tenantId-B-2`

The Amazon Chime SDK inspects inbound WebRTC media session connections for the `X-Amzn-Chime-Tenants` header and applies the following logic:

- If the header includes the session's `AppKey:tenantId`, accept the connection.
- If the header includes the session's `AppKey` but no matching `tenantId`, reject the connection with a 403 error.
- If the header does *not* include the session's `AppKey`, accept the connection.
- If the header includes the session's `AppKey`, but the session doesn't have at least one allowed `tenantId`, reject the connection with a 403 error. This may be a developer bug.
- If the header is not present, accept the connection. If users can access the application from outside the organization's network, they can also access all sessions.

HTTPS header examples

The following examples show some of the ways to use AppKeys and TenantIDs in HTTPS headers.

One app with one tenant

```
X-Amzn-Chime-App-Keys: AppKey
```

```
X-Amzn-Chime-Tenants: AppKey:orgId
```

Users can access only the organization's WebRTC media sessions in the specified app. All other apps are blocked.

One app with two tenants

```
X-Amzn-Chime-App-Keys: AppKey
```

```
X-Amzn-Chime-Tenants: AppKey:engineeringId,salesId
```

Users can access only the media sessions for engineering and sales in the specified app. All other apps are blocked.

Two Apps, One limited to a Tenant

```
X-Amzn-Chime-App-Keys: AppKey1,AppKey2
```

```
X-Amzn-Chime-Tenants: AppKey1:orgId
```

Users can access only the organization's media sessions in App 1, and any session in App 2. All other apps are blocked.

Understanding Amazon Chime SDK meeting lifecycle events

The Amazon Chime SDK sends meeting lifecycle events, which you can use to trigger notifications and initiate downstream workflows. Some examples of using meeting events include:

- Updating metadata when an attendee joins or leaves an Amazon Chime SDK meeting.
- Implementing push notifications or rosters for an Amazon Chime SDK meeting.
- Measuring the usage of video and content sharing in Amazon Chime SDK meetings.

You can send events to Amazon EventBridge, Amazon Simple Notification Service (SNS), and Amazon Simple Queue Service (SQS). For more information, refer to Events from AWS services in the *Amazon EventBridge User Guide*.

Amazon Chime SDK meeting starts

The Amazon Chime SDK sends this event when a new meeting starts.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:MeetingStarted",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK meeting ends

The Amazon Chime SDK sends this event when an active meeting ends.

> **Note**
>
> For efficiency, the service also sends this event when you call the DeleteMeeting API.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
```

```
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:MeetingEnded",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee is added

The Amazon Chime SDK sends this event when a new attendee is added to an active meeting.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeAdded",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee is deleted

The Amazon Chime SDK sends this event when you use the DeleteAttendee API to remove an attendee from an active meeting.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeDeleted",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee is authorized

The Amazon Chime SDK sends this event when a user, already joined to the meeting, uses the same join token to join the meeting again. For example, a user may switch from a desktop machine to a mobile device. This effectively "hands off" the meeting to the new device.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
```

```
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeAuthorized",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee joins a meeting

The Amazon Chime SDK sends this event when an existing attendee joins an Amazon Chime SDK meeting using the specified network transport.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeJoined",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "networkType": "Voip",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
```

```
  }
}
```

Amazon Chime SDK attendee leaves a meeting

The Amazon Chime SDK sends this event when an existing attendee leaves an Amazon Chime SDK meeting using the specified network transport.

> **Note**
>
> The service never sends `chime:AttendeeLeft` AND `chime:AttendeeDropped` events for the same "leave" action. Dropping and leaving are different actions, and the system sends the event that corresponds to each action.
> For example, say an attendee with a poor connection joins a meeting at 11 AM. You can expect the following actions:
>
> ```
> 11:00 API – CreateAttendee, CreateMeetingWithAttendee, or BatchCreateAttendee
> 11:00   Event – chime:AttendeeAdded
> 11:01 Action – user joins meeting
> 11:01   Event – chime:AttendeeJoined
> 11:02 Action – user's connection drops
> 11:02   Event – chime:AttendeeDropped
> 11:03 Action – user's connection restored
> 11:03   Event – chime:AttendeeJoined
> 11:30 Action – user leaves meeting
> 11:30   Event – chime:AttendeeLeft
> ```

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
```

```
    "version": "0",
    "eventType": "chime:AttendeeLeft",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "networkType": "Voip",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee is dropped from a meeting

The Amazon Chime SDK sends this event when a current attendee is dropped from an Amazon Chime SDK meeting, usually because of a poor connection. When the service doesn't receive packets for 10-15 seconds, it considers the cient dropped and issues the event.

The service usually triggers drop actions, but clients can also trigger them. For example, say a user switches their laptop from Wi-Fi to Ethernet. That constitutes a network adapter change, and the connection is reset. In turn, that resets the websocket and triggers a combined drop-join action.

> **Note**
>
> The service never sends `chime:AttendeeLeft` AND `chime:AttendeeDropped` events for the same "leave" action. Dropping and leaving are different actions, and the system sends the event that corresponds to each action.
> For example, say an attendee with a poor connection joins a meeting at 11 AM. You can expect the following actions:
>
> ```
> 11:00 API – CreateAttendee, CreateMeetingWithAttendee, or BatchCreateAttendee
> 11:00   Event – chime:AttendeeAdded
> 11:01 Action – user joins meeting
> 11:01   Event – chime:AttendeeJoined
> 11:02 Action – user's connection drops
> 11:02   Event – chime:AttendeeDropped
> 11:03 Action – user's connection restored
> 11:03   Event – chime:AttendeeJoined
> 11:30 Action – user leaves meeting
> 11:30   Event – chime:AttendeeLeft
> ```

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeDropped",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "networkType": "Voip",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee starts streaming video

The Amazon Chime SDK sends this event when an existing attendee starts streaming video.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeVideoStarted",
```

```
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee stops streaming video

The Amazon Chime SDK sends this event when an existing attendee stops streaming video.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeVideoStopped",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee starts sharing screen

The Amazon Chime SDK sends this event when an existing attendee starts sharing their screen.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeContentJoined",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee stops sharing screen

The Amazon Chime SDK sends this event when an existing attendee stops sharing their screen.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeContentLeft",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
```

```
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee capabilities updated

The Amazon Chime SDK sends this event when an existing attendee's capabilities are updated.

Example Event data

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "id": "12345678-1234-1234-1234-111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": [],
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeCapabilitiesUpdated",
    "success": "1", // value can be 1 or 0. 1 means success, 0 means failure
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "externalMeetingId": "mymeeting",
    "attendeeId": "attendeeId",
    "externalUserId": "externalUserId"
    "mediaRegion": "us-east-1"
    "attendeeCapabilities": {
    "audio": "SendReceive",
    "video": "SendReceive",
    "content": "SendReceive"
   }
  }
}
```

Amazon Chime SDK attendee content joins a meeting

The Amazon Chime SDK sends this event when a content share joins an Amazon Chime SDK meeting using the specified network transport.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeContentJoined",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "networkType": "Voip",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee content leaves a meeting

The Amazon Chime SDK sends this event when a content share leaves an Amazon Chime SDK meeting using the specified network transport.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
```

```
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeContentLeft",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "networkType": "Voip",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee content drops from a meeting

The Amazon Chime SDK sends this event when a content share drops from an Amazon Chime SDK meeting, typically because of low bandwidth.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeContentDropped",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "networkType": "Voip",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee content starts streaming video

The Amazon Chime SDK sends this event when a content share starts streaming video.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeContentVideoStarted",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Amazon Chime SDK attendee content stops streaming video

The Amazon Chime SDK sends this event when a content share stops streaming video.

Example Event data

The following example shows the data for this event.

```
{
  "version": "0",
  "source": "aws.chime",
  "account": "111122223333",
  "region": "us-east-1",
  "detail-type": "Chime Meeting State Change",
  "time": "yyyy-mm-ddThh:mm:ssZ",
```

```
  "resources": []
  "detail": {
    "version": "0",
    "eventType": "chime:AttendeeContentVideoStopped",
    "timestamp": 12344566754,
    "meetingId": "87654321-4321-4321-1234-111122223333",
    "attendeeId": "87654321-4321-4321-1234-111122223333",
    "externalUserId": "87654321-4321-4321-1234-111122223333",
    "externalMeetingId": "87654321-4321-4321-1234-111122223333",
    "mediaRegion": "us-east-1"
  }
}
```

Understanding Amazon CloudWatch metrics for the Amazon Chime SDK

When you use the Amazon Chime SDK, it sends service and usage metrics to CloudWatch. The metrics enable you to use CloudWatch graphs and dashboards to monitor how you consume Amazon Chime SDK services. The metrics capture data for each API that you call.

The following sections list and describe the metrics.

Topics

- Service metrics
- API usage metrics

Service metrics

The Amazon Chime SDK publishes to the following service metrics to the `AWS/ChimeSDK` namespace:

Metric	Unit	Description
`AttendeeAuthorizationSuccess`	Count	Total count of successful authorization attempts. Success means that an attendee was allowed to join the meeting.

Metric	Unit	Description
`AttendeeAuthorizationError`	Count	Total count of authorization failures, indicates that the attendee couldn't join the meeting.
`AttendeeAudioDrops`	Count	Total count of audio drops.
`AttendeeContentDrops`	Count	Total count of content share drops.
`MeetingSQSNotificationErrors`	Count	Total count of SQS Notification errors.
`MeetingSNSNotificationErrors`	Count	Total count of SNS Notification errors.

API usage metrics

The API usage metrics correspond to the AWS service quotas. You can configure alarms that alert you when your usage approaches a service quota. For more information about CloudWatch integration with service quotas, see AWS usage metrics in the *Amazon CloudWatch User Guide*.

The Amazon Chime SDK publishes the following API metrics in the `AWS/Usage` namespace, with the `ChimeSDK` service name.

Metric	Description
`CallCount`	The total number of calls made to an API in the Amazon Chime SDK. SUM represents the total number of calls to the API during the specified period.
`ErrorCount`	The total number of errors thrown by an API in the Amazon Chime SDK. SUM represents the total number of calls to the API during the specified period.

Metric	Description
`ThrottleCount`	The total number of throttling errors thrown by an API in the Amazon Chime SDK. SUM which represents the total number of calls to the API during the specified period.

The Amazon Chime SDK publishes usage metrics to the `AWS/Usage` namespace with the following dimensions:

Dimension	Description
Service	The name of the AWS service containing the resource. For Amazon Chime SDK usage metrics, the value for this dimension is `ChimeSDK`.
Type	The type of entity being reported. The only valid value for Amazon Chime SDK usage metrics is `API`.
Resource	The type of resource reporting the metric. For Amazon Chime SDK usage metrics, the value for this dimension is the name of the API.
Class	The class of resource being tracked. The only valid value for Amazon Chime SDK metrics is None.

Creating Amazon Chime SDK media pipelines

Important

You and your end users must understand that recording Amazon Chime SDK meetings may be subject to laws or regulations regarding the recording of electronic communications. It is your and your end users' responsibility to comply with all applicable laws regarding

the recordings, including properly notifying all participants in a recorded session that the session or communication is being recorded, and obtain their consent.
You and your end users are responsible for all content streaming using the media live connector service, and must ensure that such content does not violate the law, infringe or misappropriate the rights of any third party, or otherwise violate a material term of your agreement with Amazon.

To capture or stream an Amazon Chime SDK meeting, you create media pipelines. A media pipeline can consist of one of these pipelines:

- **Media capture** – You use media capture pipelines to capture audio, video, and content share streams, plus meeting events and data messages. All media capture pipelines save their data to Amazon Simple Storage Service (S3) bucket that you create. You can create one media capture pipeline per Amazon Chime SDK meeting. For more information, refer to Understanding Amazon Chime SDK media capture pipeline creation, later in this section.
- **Media concatenation** – You use media concatenation pipelines to concatenate the artifacts from a media capture pipeline. Concatenation pipelines work independently of media capture and live connector pipelines. For more information, refer to Creating media concatenation pipelines, later in this section.
- **Media live connector** – You use media live connector pipelines to connect to services that enable you to stream Amazon Chime SDK meetings to an RTMP endpoint. You can create up to one media live connector pipeline per Amazon Chime SDK meeting. For more information, refer to Creating media live connector pipelines, later in this section.
- **Media stream** – You use media stream pipelines to capture individual audio for all the attendees in a meeting, plus the mixed audio generated by a media concatenation pipeline. All media stream pipelines save their data to Amazon Kinesis Video Streams (KVS). For more information, refer to Creating media stream pipelines, later in this section.

The pipelines you create depend on the namespace you use. If you use the `Chime` namespace, you can only create media capture pipelines. If you use the `ChimeSdkMediaPipelines` namespace, you can also create media concatenation and media live connector pipelines, and use compositing features. If you want to migrate to the `ChimeSdkMediaPipelines` namespace, refer to Migrating to the ChimeSdkMediaPipelines namespace.

The following table lists the default limits for active media pipelines in each Region. Each type of pipeline counts toward the limit.

Region	Default active pipeline limit
us-east-1	100
us-west-2	10
ap-northeast-1	10
ap-northeast-2	10
ap-south-1	10
ap-southeast-1	10
ap-southeast-2	10
ca-central-1	10
eu-central-1	10
eu-west-2	10

Note

If you exceed the limit for any Region, the CreateMediaCapturePipeline, CreateMediaConcatenationPipeline, and CreateMediaLiveConnectorPipeline APIs will throw **Resource Limit Exceeded** exceptions.
You can use the **Service Quotas** page in the AWS console to adjust your active pipeline limits, or you can contact your customer support representative. For more information about the Amazon Chime SDK meeting limits, see Quotas for the Amazon Chime SDK.

Before you begin, you must integrate your client application with the Amazon Chime SDK client library. For more information, see Learn about the Amazon Chime SDK client libraries. For more information about media pipelines, see Capture Amazon Chime SDK Meetings Using media pipelines.

Topics

Migrating to the ChimeSdkMediaPipelines namespace

You use the `ChimeSdkMediaPipelines` namespace to address media pipeline API endpoints in any AWS Region in which they're available. Use this namespace if you're just starting to use the Amazon Chime SDK. For more information about Regions, refer to Available AWS Regions for the Amazon Chime SDK service in this guide.

Existing applications that use the Amazon Chime namespace should plan to migrate to the dedicated namespace.

Topics

Reasons to migrate your pipelines

We encourage you to migrate to the `ChimeSdkMediaPipelines` namespace for these reasons:

Choice of API Endpoint

The Amazon Chime SDK Media Capture namespace is the only API namespace which can use API endpoints in any Region that makes them available. For more information about Regions, refer

to Available AWS Regions for the Amazon Chime SDK service. If you want to use API endpoints other than `us-east-1`, you must use the `ChimeSdkMediaPipelines` namespace. For more information about the current endpoints, refer to API mapping in this guide.

Updated and new media pipeline APIs

We only add or update media pipeline APIs in the `ChimeSdkMediaPipelines` namespace.

Before you migrate your pipelines

Before you migrate, be aware of the differences between the namespaces. The following table lists and describes them.

Item	Media pipelines namespace	Chime namespace
Namespace names	ChimeSdkMediaPipelines	Chime
Regions	Multiple	us-east-1 only
Endpoints	https://media-pipelines-chime.*region*.amazonaws.com	https://service.chime.aws.amazon.com
Service principal	mediapipelines.chime.amazonaws.com	chime.amazonaws.com
APIs	Only APIs for media pipelines	APIs for media pipelines and other parts of Amazon Chime
Meetings	Media pipelines in the `us-west-2` , `ap-southeast-1` , and `eu-central-1` regions only work with meetings created in the Amazon Chime SDK Meetings namespace. Media pipelines in the `us-east-1` region work with meetings created by any meeting endpoint in either namespace.	Media pipelines work with meetings created by any meetings endpoint in either namespace.

Item	Media pipelines namespace	Chime namespace
Default active media pipelines	100 in the us-east-1 Region, and 10 in the us-west-2, ap-southeast-1, and eu-central-1 Regions.	100 in us-east-1 only.
Service-linked role	AWSServiceRoleForAmazonChimeSDKMediaPipelines	
Tags	Available	Not available for the media pipeline APIs.
CloudTrail event source	chime-sdk-media-pipelines.amazonaws.com	chime.amazonaws.com.
Media live connector	Available	Not available for the media pipeline APIs.
Compositing	Available	Not available for the media pipeline APIs.
Concatenation	Available	Not available.

The following list provides more information about the differences between the Chime and AWSChimeSdkMediaPipelines namespaces.

Namespace names

The Amazon Chime SDK namespace uses the `AWS.Chime` formal name. The Amazon Chime SDK Media Pipelines namespace uses the `AWS.ChimeSDKMediaPipelines` formal name. The precise format of the name varies by platform.

For example, this line of Node.js code addresses the `chime` namespace:

```
const chimeMediaPipelines = AWS.Chime();
```

To migrate to the Media Pipelines SDK namespace, update that code with the new namespace and the endpoint region.

```
const chimeMediaPipelines = AWS.ChimeSDKMediaPipelines({ region: "eu-central-1" });
```

Regions

The Amazon Chime namespace only addresses API endpoints in the US-EAST-1 region. The Amazon Chime SDK Media Pipelines namespace addresses Amazon Chime SDK media pipeline API endpoints in any Region that has them. For a current list of media pipeline Regions, see Available AWS Regions for the Amazon Chime SDK service in this guide.

Endpoints

To modify a media capture pipeline, you must use the same endpoint that you created the pipeline in. For example, if you created pipelines via an endpoint in eu-central-1, you must use eu-central-1 to interact with that pipeline. For more information about the current endpoints, refer to API mapping in this guide.

Service principal

The Amazon Chime SDK Media Pipelines namespace uses a new service principal: `mediapipelines.chime.amazonaws.com`. If you have Amazon S3 bucket or other IAM policies that grant access to services, you need to update those polices to grant access to the new service principal.

For example, when you create a media pipelines, you must add the policy permissions listed in Creating an Amazon S3 bucket to the new service principal. For more information about policies, see AWS JSON policy elements: Principal in the IAM User Guide.

APIs

The Amazon Chime SDK Media Pipelines namespace only contains APIs that create and manage media pipelines. The Amazon Chime namespace includes APIs for media pipelines, meetings, and other parts of the Amazon Chime service.

Meetings

Media pipelines in the IAD region work with meetings created by any meetings endpoint with either namespace.

Service-linked role

Only for the Amazon Chime SDK Media Pipelines namespace. Create the *AWSServiceRoleForAmazonChimeSDKMediaPipelines* role.

Tags

The Amazon Chime SDK Media Pipelines namespace supports tags. The role must have permission to call the `TagResource` operation when calling the CreateMediaCapturePipeline or CreateMediaLiveConnectorPipeline APIs with one or more tags.

Understanding Amazon Chime SDK media capture pipeline creation

You follow a multi-step process to create an Amazon Chime SDK media pipeline, and you can create several types of pipelines. The following list outlines the creation process and provide links to more information about creating the various types of pipelines.

- Create an Amazon S3 bucket. You must create the bucket in the same AWS Region as the meeting. For more information, refer to Creating an Amazon S3 bucket.
- Create a service-linked role named `AWSServiceRoleForAmazonChimeSDKMediaPipelines`. This allows media pipelines to access meetings on your behalf. For more information, refer to Creating a service-linked role for media pipelines.
- Create an IAM role with sufficient permission to interact with the Amazon Chime SDK media pipeline APIs . To create that role, we recommend adding the AmazonChimeSDK managed policy from the IAM console. The policy contains the necessary APIs.

  Your IAM role must also have permission to call the Amazon S3 GetBucketPolicy API on all resources. The following example shows a typical policy for doing so.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "s3:GetBucketPolicy",
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Once you have those items, see these topics for information on creating pipelines.

- Creating media capture pipelines
- Creating media concatenation pipelines
- Creating media live connector pipelines
- Creating media stream pipelines

Creating media capture pipelines

Media capture pipelines capture audio, video, and content share streams, plus meeting events and data messages. All media capture pipelines save their data to an Amazon Simple Storage Service (S3) bucket that you create. You can create one media capture pipeline per Amazon Chime SDK meeting.

The following sections explain how to create a media capture pipeline. Follow them in the order listed.

Topics

Creating an Amazon S3 bucket

You can use the Amazon S3 console the AWS SDKs, or the AWS CLI to to create an Amazon S3 bucket. For more information, refer to Creating a bucket, in the *Amazon Simple Storage Service (S3) User Guide*.

The Amazon S3 bucket for your media capture pipeline must belong to the same AWS account as the Amazon Chime SDK meeting. In addition, you must give the `s3:PutObject` and `s3:PutObjectAcl` permission to the Amazon Chime SDK service principal mediapipelines.chime.amazonaws.com. You can do that with the Amazon S3 console or the AWS Command Line Interface (AWS CLI). The Amazon S3 bucket must belong to one of the available Amazon Chime SDK media Regions.

> **Note**
>
> Make sure to add a policy to your IAM user to grant access to your bucket. Also, if you use a Region that AWS disables by default, you must have an Amazon S3 bucket in that Region. By default, AWS disables the following Regions, and you can't host meeting resources in them until you enable them:
>
> - Africa (Cape Town)
> - Asia Pacific (Hong Kong)
> - Asia Pacific (Jakarta)
> - Europe (Milan)
> - Middle East (Bahrain)
>
> If you use one of those Regions, it must have an Amazon S3 bucket. This applies even if you use the Amazon S3 APIs to communicate with Regions that aren't blocked by default and already have a bucket. For more information about enabling blocked regions, refer to Managing AWS Regions in the *AWS General Reference*.

Once you create a bucket, record its ARN. You use it to create a media capture pipeline.

The following example shows an Amazon S3 bucket policy.

```
{
    "Version": "2012-10-17",
    "Id": "AWSChimeMediaCaptureBucketPolicy",
    "Statement": [
        {
            "Sid": "AWSChimeMediaCaptureBucketPolicy",
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "mediapipelines.chime.amazonaws.com"
            },
            "Action": [ "s3:PutObject", "s3:PutObjectAcl" ],
            "Resource": "arn:aws:s3:::Bucket_Name/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "Account_Id"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:chime:*:Account_Id:*"
                }
            }
        }
    ]
}
```

Enabling server-side encryption for an Amazon S3 bucket

To enable server-side encryption for an Amazon Simple Storage Service (Amazon S3) bucket, you can use these types of encryption keys:

- An Amazon S3 managed key
- A customer managed key in the AWS Key Management Service (KMS)

> **Note**
>
> The Key Management Service supports two types of keys, customer managed keys and AWS managed keys. Amazon Chime SDK meetings only support customer managed keys.

Using an Amazon S3 managed key

You use the Amazon S3 console, CLI, or REST API to enable server-side encryption for an Amazon S3 bucket. In both cases, choose **Amazon S3 Key** as encryption key type. No further action is needed. When you use the bucket for media capture, the artifacts are uploaded and encrypted on server side. For more information, refer to Specifying Amazon S3 encryption in the *Amazon S3 User Guide*.

Using a key that you own

To enable encryption with a key that you manage, you need to enable the Amazon S3 bucket's server side encryption with a Customer Managed Key, then add a statement to the key policy that allows Amazon Chime to use the key and encrypt any uploaded artifacts.

1. Create a Customer Managed Key in KMS. For information about doing so, see Specifying server-side encryption with AWS KMS (SSE-KMS) in the *Amazon S3 User Guide*.
2. Add a statement to the key policy that allows the `GenerateDataKey` action to generate a key for use by the Amazon Chime SDK service principal, `mediapipelines.chime.amazonaws.com`.

   This example shows a typical statement.

```
...
{
    "Sid": "MediaPipelineSSEKMS",
    "Effect": "Allow",
    "Principal": {
        "Service": "mediapipelines.chime.amazonaws.com"
    },
    "Action": "kms:GenerateDataKey",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
           "aws:SourceAccount": "Account_Id"
        },
        "ArnLike": {
            "aws:SourceArn": "arn:aws:chime:*:Account_Id:*"
        }
    }
}
...
```

3. If you use a media concatenation pipeline, add a statement to the key policy that allows the Amazon Chime SDK service principal, `mediapipelines.chime.amazonaws.com`, to use the `kms:Decrypt` action.
4. Configure the Amazon S3 bucket to enable server side encryption with the key.

Creating the media capture pipeline

After you create and configure your Amazon S3 bucket or buckets, you create a media capture pipeline.

To create a media capture pipeline

- Call the CreateMediaCapturePipeline API.

  Use the bucket ARN as the `SinkArn` parameter.

Once successful, the Amazon Chime SDK creates an attendee that joins and captures the meeting.

After you create a media capture pipeline and set its permissions, you create a media concatenation pipeline to concatenate the 5-second media chunks into a single file. For more information, refer to Creating media concatenation pipelines, later in this section.

Working with media capture artifacts

During an Amazon Chime SDK meeting, a media capture pipeline creates the following types of artifacts.

- Audio
- Video
- Data channel messages
- Meeting events
- Transcription messages

The pipeline creates the artifacts in a set of folders in your Amazon S3 bucket, and you can configure the audio and video folders to limit certain types of artifacts. The following sections explain the folder structure, how to configure folders, how to set permissions for your Amazon S3 bucket, and how to concatenate the artifact files.

Configuring the audio folder

The audio folder contains 5-second MP4 files of the mixed audio stream, meaning they contain audio from all attendees, plus the active speaker's video. The folder contains files for the entire meeting. As desired, you can configure the folder to contain just the audio artifacts. Each file name

contains a *yyyy-mm-dd-hour-min-seconds-milleseconds* timestamp. The timestamp is in UTC, and it marks the start time. You can configure the folder to only contain audio artifacts.

```
"ArtifactsConfiguration": {
         "Audio": {
            "MuxType": "AudioOnly"
         },
         "Content": {
            "State": "Disabled"
         },
         "Video": {
            "State": "Disabled"
         }
      }
```

Configuring the video folder

The video folder contains 5-second MP4 files that contain video streams, plus content share streams if they're specified in the API request. Each file name contain a <yyyy-mm-dd-hour-min-seconds-milleseconds>-<attendeeID> timestamp with an attendee ID. The content share video chunk is appended as <yyyy-mm-dd-hour-min-seconds-milleseconds>-<attendeeID>#content.mp4. You can configure the folder to only contain video artifacts.

```
"ArtifactsConfiguration": {
         "Audio": {
            "MuxType": "AudioOnly"
         },
         "Content": {
            "State": "Disabled"
         },
         "Video": {
            "MuxType": "VideoOnly"
            "State": "Enabled"
         }
      }
```

Understanding messages in the data-channel folder

The data-channel folder contains data messages in the .txt format, and each message is a JSON object. Messages are visible with all configurations options. File names contain the *yyyy-mm-dd-hour-min-seconds-milleseconds* timestamp. This example shows the data fields in a message.

```
{
    "Timestamp": "string",
    "Topic": "string",
    "Data": "string",
    "SenderAttendeeId": "string"
}
```

Understanding the Amazon S3 bucket folder structure

The Amazon S3 buckets for media capture pipelines use this folder structure.

```
S3 bucket path/
  audio
  video
  data-channel
  meeting-events
  transcription-messages
```

Understanding meeting event files

The meeting-events folder contains meeting events in the .txt format, and each event is a JSON object. Messages are visible with all configurations options. File names contain the <yyyy-mm-dd-hour-min-seconds-milleseconds> timestamp. This example shows the fields and data in a typical event file.

```
{
    "Timestamp": "string",
    "EventType": "AttendeeJoined | AttendeeLeft | AttendeeVideoJoined |
 AttendeeVideoLeft | ActiveSpeaker | CaptureStarted | CaptureEnded  | AudioTrackMute |
 AudioTrackUnmute",
    "EventParameters": {
        # ...
    }
}
```

Understanding transcription files

The transcription-messages folder contains transcription files in the .txt format. However, the folder only receives files when you enable live transcription. For more information about enabling live transcription, see Using Amazon Chime SDK live transcription.

The folder includes all partial and complete transcription messages, and each message is a JSON object. File names contain the <yyyy-mm-dd-hour-min-seconds-milleseconds> timestamp. You can see transcription file examples at Processing a received transcript event.

Concatenating data streams

> **Note**
>
> To automate the process of concatenating media capture artifacts, refer to Creating media concatenation pipelines in this guide.

This example uses ffmpeg to concatenate video or audio files into a single mp4 file. First, create a filelist.txt file that contains all the input files. Use this format:

```
file 'input1.mp4'
file 'input2.mp4'
file 'input3.mp4'
```

Next, use this command to concatenate the input file:

```
ffmpeg -f concat -i filelist.txt -c copy output.mp4
```

For more information about media concatenation pipelines, refer to Creating media concatenation pipelines in this guide.

Creating media concatenation pipelines

You use media concatenation pipelines to concatenate the artifacts (files) generated by media capture pipelines.

Media capture pipelines capture the contents of a meeting by chunking the media streams and storing those artifacts in your Amazon S3 bucket. Media capture pipelines create the following types of artifacts:

- Audio
- Video
- Content shares

- Data channel messages
- Transcription messages
- Meeting events
- Composited video, meaning content shares and multiple video streams displayed in a grid as video tiles.

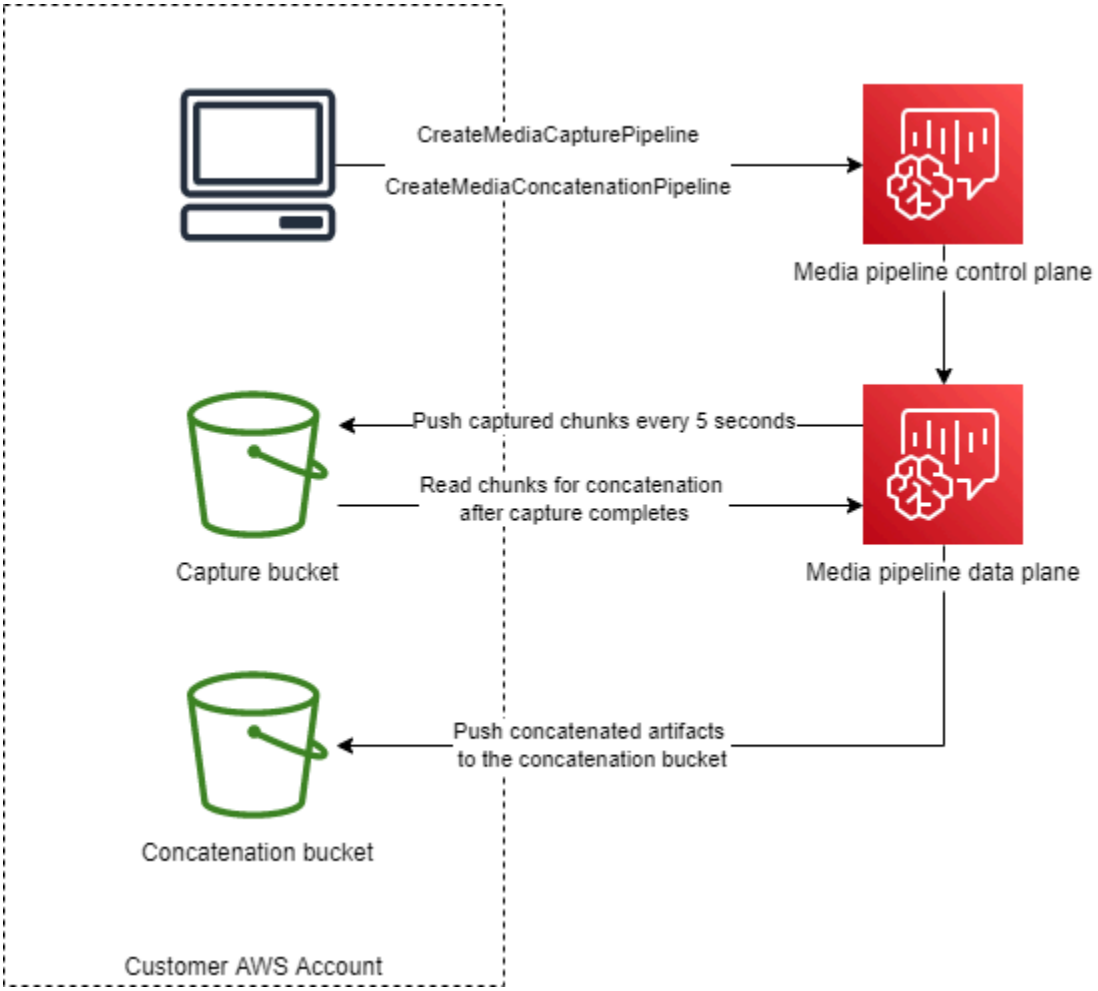
Media concatenation pipelines allow you to concatenate each type of artifact into a single file, and then store those larger files in your Amazon S3 bucket. You can create a media concatenation pipeline without waiting for the media capture event to end, but the concatenation pipeline only starts concatenating when the capture pipeline stops.

> **Note**
>
> Media capture pipelines, media concatenation pipelines, and Amazon S3 buckets must reside in the same AWS account.

Topics

Concatenation pipeline architecture

The following diagram shows the architecture of a media concatenation pipeline.



In the diagram, on receiving a CreateMediaCapturePipeline request, the media pipeline control plane starts a media capture pipeline in the media pipeline data plane. The data plane then pushes captured chunks to the capture bucket every 5 seconds. On receiving a CreateMediaConcatenationPipeline request, the media pipeline control plane waits for the specified media capture pipeline to finish, then starts a media concatenation pipeline in the media pipeline data plane. The data plane then reads the captured chunks in the bucket and pushes the concatenated artifacts to the concatenation bucket.

Building a media concatenation pipeline

You follow a multi-step process to create an Amazon Chime SDK media concatenation pipeline. The following steps describe the process.

1. Create an Amazon S3 bucket for use as the media capture pipeline's data sink, and then configure the bucket policy. For information about enabling server-side encryption for the Amazon S3 bucket, refer to Enabling server-side encryption for an Amazon Amazon S3 bucket

in this guide. If you created an Amazon Amazon S3 bucket for use with media capture pipelines, you must add the `s3:GetObject` and `s3:ListBucket` actions to that bucket's policy. The `s3:ListBucket` action requires permission on the bucket. The other actions require permission on the objects in the bucket. You must use two different Amazon Resource Names (ARNs) to specify bucket-level and object-level permissions.

The following example shows the bucket policy. Copy and paste this example as needed.

```
{
    "Version": "2012-10-17",
    "Id": "AWSChimeMediaCaptureBucketPolicy",
    "Statement": [
        {
            "Sid": "AWSChimeMediaCaptureBucketPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "mediapipelines.chime.amazonaws.com"
                ]
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::[Bucket-Name]/*",
                "arn:aws:s3:::[Bucket-Name]"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "[Account-Id]"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:chime:*:[Account-Id]:*"
                }
            }
        }
    ]
}
```

2. Create an Amazon Amazon S3 bucket for use as the media concatenation pipeline's data sink, and then configure the bucket policy. For information about enabling server-side encryption for the Amazon S3 bucket, refer to Enabling server-side encryption for an Amazon Amazon S3 bucket in this guide.

   The following example shows the policy.

```
{
    "Version": "2012-10-17",
    "Id": "AWSChimeMediaConcatenationBucketPolicy",
    "Statement": [
        {
            "Sid": " AWSChimeMediaConcatenationBucketPolicy ",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "mediapipelines.chime.amazonaws.com"
                ]
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::[Bucket-Name]/*",
                "arn:aws:s3:::[Bucket-Name]"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "[Account-Id]"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:chime:*:[Account-Id]:*"
                }
            }
        }
    ]
}
```

> **Note**
>
> You can use a single Amazon S3 bucket for media capture and media concatenation pipelines. However, if you do that, you must add the `s3:GetObject` and `s3:ListBucket` permissions to the media concatenation bucket policy shown in step 2. If you don't want the concatenation bucket policy to have those permissions, then create separate buckets for each pipeline.

3. Use the CreateMediaCapturePipeline API to create a media capture pipeline. As part of that, get the pipeline's ARN. For information about getting the ARN, refer to Understanding Amazon Chime SDK media capture pipeline creation. You use the ARN in the next step.
4. Use the CreateMediaConcatenationPipeline API to create a concatenation pipeline.

   The following example shows a request body. The *`Path`* field is optional, and it defaults to the concatenation pipeline's ID.

   > **Note**
   >
   > You must use a `MediaPipelineArn` created within the last 30 days.

```
{
    "Sources": [
        {
            "Type": "MediaCapturePipeline",
            "MediaCapturePipelineSourceConfiguration": {
                "MediaPipelineArn": "Media_Pipeline_Arn",  //must be <30 days old
                "ChimeSdkMeetingConfiguration": {
                    "ArtifactsConfiguration": {
                        "Audio": {
                            "State": "Enabled"
                        },
                        "Video": {
                            "State": "Enabled | Disabled"
                        },
                        "Content": {
                            "State": "Enabled | Disabled"
                        },
                        "DataChannel": {
```

```
                            "State": "Enabled | Disabled"
                        },
                        "TranscriptionMessages": {
                            "State": "Enabled | Disabled"
                        },
                        "MeetingEvents": {
                            "State": "Enabled | Disabled"
                        },
                        "CompositedVideo": {
                            "State": "Enabled | Disabled"
                        }
                    }
                }
            }
        }
    ],
    "Sinks": [
        {
            "Type": "S3Bucket",
            "S3BucketSinkConfiguration": {
                "Destination": "arn:aws:s3:::[Bucket_Name]/[Path]"
            }
        }
    ]
}
```

Concatenation starts whenever the capture pipeline stops. The concatenation pipeline stops after completing the concatenation.

Understanding the Amazon S3 bucket folder structure

The Amazon S3 buckets for media concatenation pipelines use this folder structure:

```
S3 bucket path/
  audio
  video
  composited-video
  data-channel
  meeting-events
  transcription-messages
```

> **Note**
>
> If you specify a prefix when you create a media pipeline, the path to the folders becomes *bucket name/prefix*. Without a prefix, the path becomes *bucket name/media pipeline ID*. You specify a prefix in the `Destination` field of the `S3BucketSinkConfiguration` object. The concatenated file names consist of *media pipeline ID*.mp4 for media files and *media pipeline ID*.txt for text files.

Creating media live connector pipelines

The following sections list and describe the Real-Time Messaging Protocol (RTMP), audio, and video settings for a media live connector pipeline.

RTMP settings

Media live connector pipelines support RTMP over a TLS/SSL connection. The sink URL consists of the stream URL and stream key. The URLs follow this format:

```
rtmp(s)://stream-server/stream-key
```

The following examples show how to connect to common streaming platforms.

- **Amazon Interactive Video Service (IVS)** – rtmps://a1b2c3d4e5f6.global-contribute.live-video.net:443/app/*IVS-stream-key*
- **YouTube** – rtmps://a.youtube.com/live2/*stream-key*
- **Twitch** – rtmps://live.twitch.tv/app/*primary-stream-key*

> **Important**
>
> RTMPS uses encryption to help ensure that a stream is not intercepted by an unauthorized entity. As a best practice, use RTMPS when you need additional data security.

Audio settings

Media live connector pipelines support the following audio settings:

- **Codec** – AAC
- **Sample rate** – 44100 Hz or 48000 Hz. The default is 44100Hz.
- **Channels** – Mono or stereo. The default is mono.

Video settings

Media live connector pipelines use the H264 encoder. You can use HD at 1280x720 or FHD at 1920x1080. Both resolutions use 30 frames per second, with a keyframe every two seconds.

Stopping media live connector pipelines

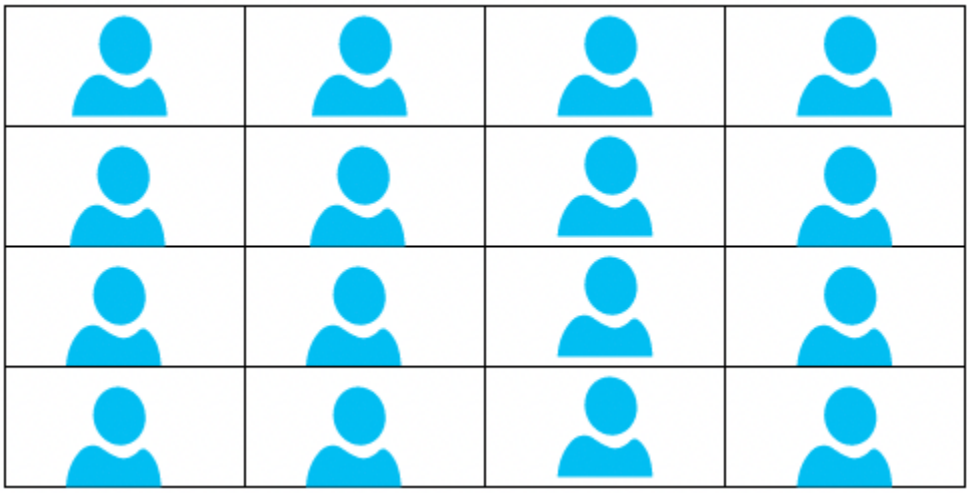
As a best practice for stopping media live connector pipelines, call the DeleteMediaPipeline API. Ending a stream on a streaming platform such as IVS does not stop a media live connector pipeline.

Compositing audio and video into a single view

The Amazon Chime SDK media pipelines support compositing of audio, webcam videos, and content-share video streams into a single view. You can then use live connector to send that single view to streaming services such as Amazon Interactive Video Service, Twitch, or YouTube Live. Composited video can also be captured to Amazon Simple Storage Service for storage or further consumption.

Compositing uses a default screen layout called `GridView`, which has the following behaviors.

- When only webcam videos are active, `GridView` organizes the streams in the following grid pattern:



The grid displays a maximum of 25 webcam streams, and it orders the tiles by when users turn on their cameras.

- `GridView` provides two canvas orientations, `Landscape` and `Portrait`. Landscape, the default orientation, supports video resolutions of 1280x720 and 1920x1080 for FHD. Portrait supports resolutions of 720x1280 and 1080x1920 for FHD.
- You can configure the order, position, total number, tile aspect ratio, corner radius, border color, border thickness and highlight color of the video tiles.
- During a meeting, when someone shares their screen, the webcam video tiles transition dynamically to make room for the content share. You control those transitions, and the locations of the video tiles, by using one of the layout configurations described in the next section.

About the layout configurations

When someone starts a content share, you can choose how to composite the content share and webcam video streams by using one of the following layout configurations.

- `ActiveSpeakerOnlyConfiguration` composites the content video full screen, with the active speaker's webcam video overlaid in a corner. You can specify the corner.
- `PresenterOnlyConfiguration` composites the content video full screen, with the presenter's webcam video overlaid in a corner. You can specify the corner.
- `VerticalLayoutConfiguration` composites the content video with the webcam video in an adjacent vertical column. You can display the column to the right or left of the content share.
- `HorizontalLayoutConfiguration` composites the content video with the webcam video in an adjacent horizontal row. You can display the row above or below the content share.

Composited layouts automatically transition between `GridView` and your chosen layout, based on whether content share is active or not.

The following topics explain how to use the global `GridView` settings and each configuration layout.

Topics

- Using the layout configurations

Setting canvas orientation

In compositing, the *canvas* contains all your video streams. You can specify a `Landscape` or `Portrait` orientation for the canvas. Landscape provides a 16:9 aspect ratio. Portrait provides a 9:16 aspect ratio.

The following image shows the portrait orientation.

The following example shows how to implement a portrait canvas with the video tile in the upper-right corner. In this example, the active speaker appears in the tile. For more information, see ActiveSpeakerOnlyConfiguration

```
{
   "CompositedVideo":{
      "Layout":"GridView",
      "Resolution":"FHD",
      "GridViewConfiguration":{
         "ContentShareLayout":"ActiveSpeakerOnly",
         "ActiveSpeakerOnlyConfiguration":{
```

```
            "ActiveSpeakerPosition":"TopRight"
         }
      },
      "CanvasOrientation": "Portrait"
   }
}
```

The following image shows the landscape orientation.

CanvasOrientation

Description – The orientation setting, Landscape or Portrait.

Allowed values – `Landscape` | `Portrait`

Required – No

Default – Landscape

Setting border and corner attributes

As desired, you can use the `VideoAttribute` parameter to specify border and corner settings for your video tiles. You can specify colors, widths, and rounded corners. You can also specify a highlight color, and the border changes to that color when someone speaks.

Your attribute settings apply to all layouts, regardless of content sharing.

The following image shows a video tile with a border color and corner radius applied.

The following example shows how to use each attribute. In this case, video tiles have rounded corners with a five-pixel radius. The tiles have a green border, also five pixels wide. When the speaker talks, the `HighlightColor` attribute changes the border color to red.

```
{
   "CompositedVideo":{
      "Layout":"GridView",
      "Resolution":"FHD",
      "GridViewConfiguration":{
         "ContentShareLayout":"ActiveSpeakerOnly",
         "ActiveSpeakerOnlyConfiguration":{
            "ActiveSpeakerPosition":"TopRight"
         }
         "VideoAttribute": {
            "CornerRadius"   : 10,
            "BorderColor"    : "Green",
            "HighlightColor" : "Red",
            "BorderThickness": 5
          },
       }
```

```
    }
}
```

VideoAttribute

Description – Specifies the settings for video tile borders and rounded corners

Allowed values – `BorderColor` | `BorderThickness` | `CornerRadius` | `HighlightColor`

Required – No

VideoAttribute.BorderColor

Description – Defines the border color for all video tiles

Allowed values – Color names, such as Red, Green, or Blue

Required – No

VideoAttribute.BorderThickness

Description – Defines the border thickness in pixels for all video tiles

Type – Integer

Allowed values – 1–20

Required – No

VideoAttribute.CornerRadius

Description – Defines the corner radius in pixels for all video tiles.

Type – Integer

Allowed values – 1–20

Required – No

VideoAttribute.HighlightColor

Description – Defines a border color that appears when a presenter or speaker talks

Allowed values – Color names, such as Red, Green, or Blue

Required – No

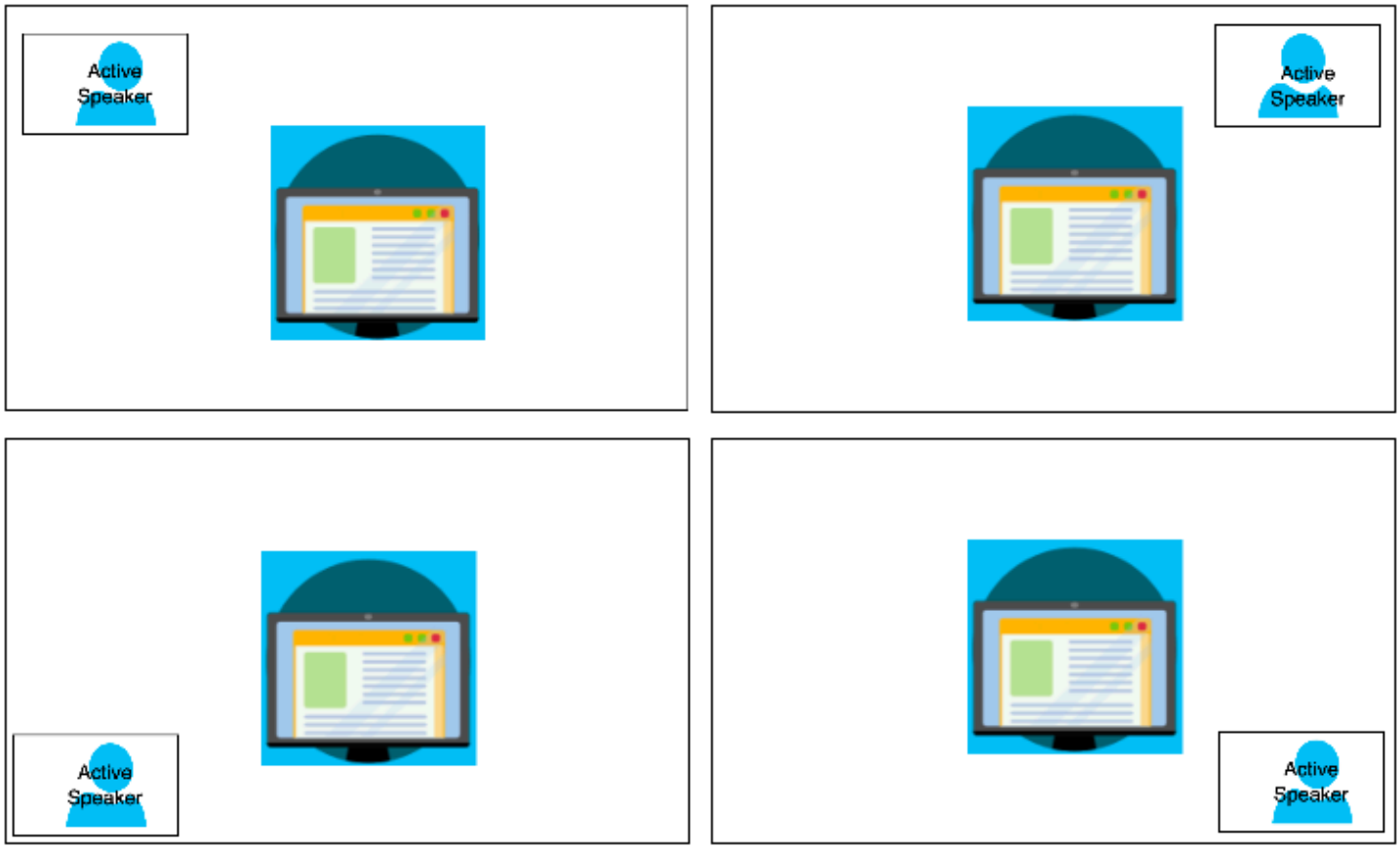
Using the layout configurations

The following topics explain how to use the different configuration layouts. The layouts only take effect when someone starts a content share. Expand each section to learn more.

ActiveSpeakerOnlyConfiguration

`ActiveSpeakerOnlyConfiguration` displays the content share and the active speaker's video, meaning the person talking appears in the small video tile that overlays the content share stream.

The following image shows the configuration and the available locations for the speaker tile.



The following example shows how to implement the `ActiveSpeakerOnly` layout programmatically. In this case, the presenter tile appears in the upper-left corner.

```
{
   "CompositedVideo":{
```

```
      "Layout":"GridView",
      "Resolution":"FHD",
      "GridViewConfiguration":{
         "ContentShareLayout":"ActiveSpeakerOnly",
         "ActiveSpeakerOnlyConfiguration":{
            "ActiveSpeakerPosition":"TopLeft"
         }
      }
   }
}
```

ActiveSpeakerOnlyConfiguration

Description – The configuration settings for an `ActiveSpeakerOnly` video tile

Type – ActiveSpeakerOnlyConfiguration object

Required – No

ActiveSpeakerOnlyConfiguration.ActiveSpeakerPosition

Description – The position of the active speaker video tile

Type – String

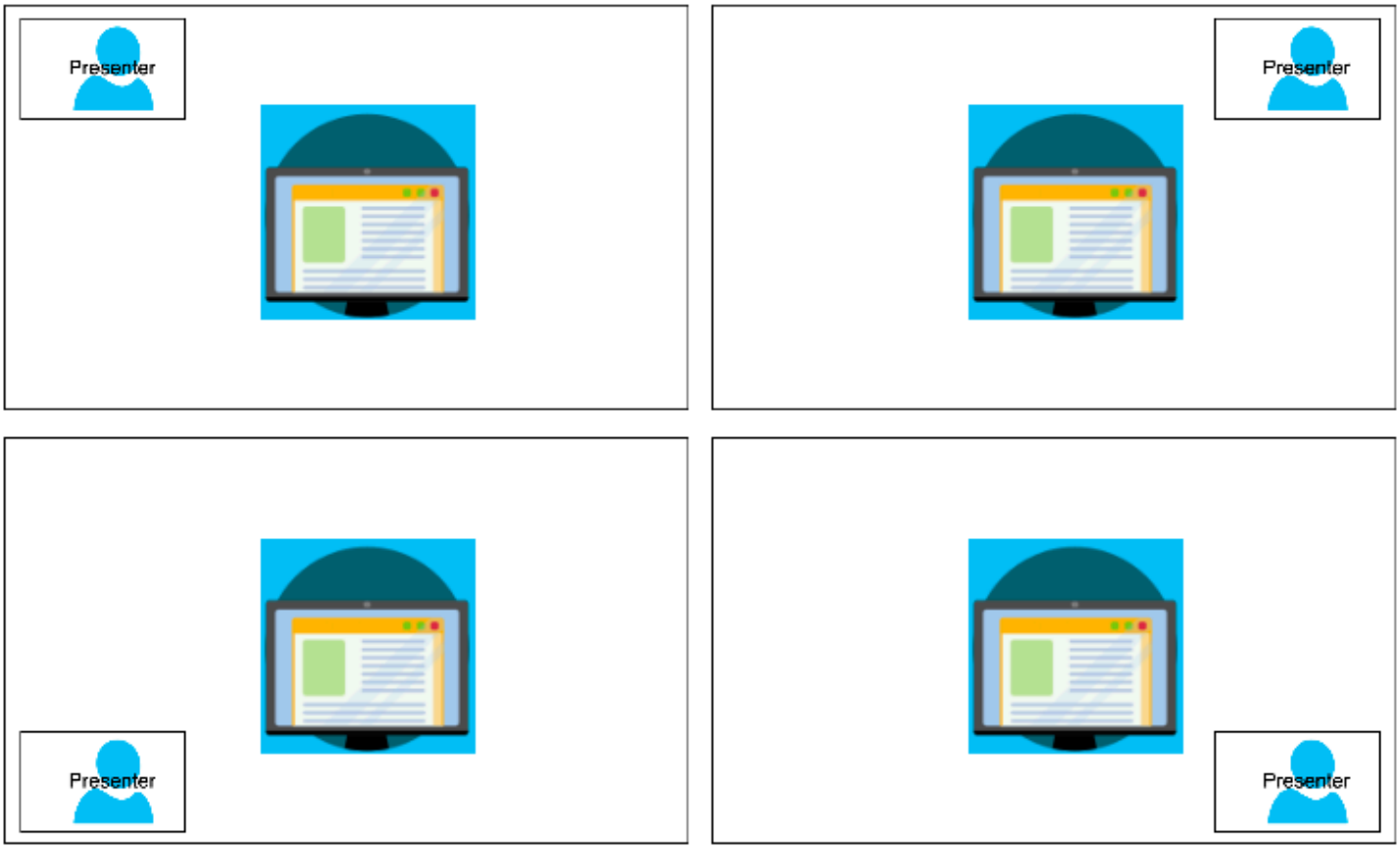
Valid values – `TopLeft` | `TopRight` | `BottomLeft` | `BottomRight`

Required – No

Default – `TopRight`

PresenterOnlyConfiguration

`PresenterOnlyConfiguration` displays the content share and only the presenter's video regardless of who talks. The following image shows the configuration.



The following example shows how to implement the layout programmatically with the presenter at top-right.

```
{
    "CompositedVideo": {
    "Layout": "GridView",
    "Resolution": "FHD",
     "GridViewConfiguration": {
         "ContentShareLayout": "PresenterOnly",
         "PresenterOnlyConfiguration": {
             "PresenterPosition": "TopRight"
             }
         }
     }
}
```

PresenterOnlyConfiguration

Description – The configuration settings for a `PresenterOnly` layout

Type – `PresenterOnlyConfiguration` object

Required – No

PresenterOnlyConfiguration.PresenterPosition

Description – The position of the presenter video tile

Type – String

Valid values – `TopLeft` | `TopRight` | `BottomLeft` | `BottomRight`

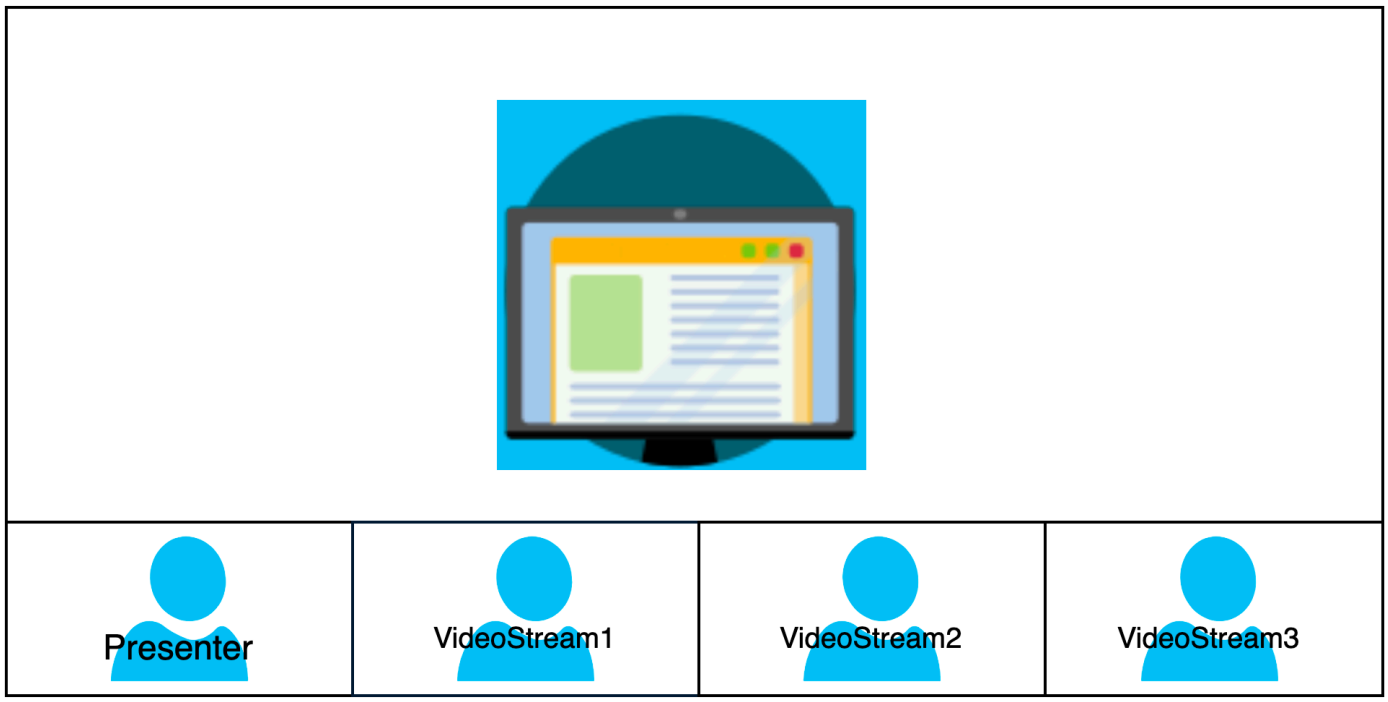
Required – No

Default – `TopRight`

HorizontalLayoutConfiguration

`HorizontalLayoutConfiguration` displays the content sharing and video streams horizontally. You can use the `TilePosition` setting to display the tiles above or below the content share stream. Presenters always appear on the left. Additional tiles appear in the order dictated by `JoinSequence`.

The following image shows the tiles below the content share stream.



The following example shows how to implement a horizontal layout programmatically. In this case, the layout orders the tiles by `SpeakerSequence` and places them below the screen share. The layout allows a maximum of four tiles and applies a 16/9 aspect ratio.

```
{
   "CompositedVideo":{
      "Layout":"GridView",
      "Resolution":"FHD",
      "GridViewConfiguration":{
         "ContentShareLayout":"Horizontal",
         "HorizontalLayoutConfiguration":{
            "TileOrder":"SpeakerSequence",
            "TilePosition":"Bottom",
            "TileCount":4,
            "TileAspectRatio":"16/9"
         }
      }
   }
}
```

HorizontalLayoutConfiguration

Description – The configuration settings for a horizontal layout

Type – HorizontalLayoutConfiguration object

Required – No

HorizontalLayoutConfiguration.TilePosition

Description – Places tiles above or below a content share.

Type - String

Valid values – `Bottom` | `Top`

Required – No

Default – Bottom

HorizontalLayoutConfiguration.TileOrder

Description – Orders tiles by when users join or when they speak

Type – String

Valid values – `JoinSequence` | `SpeakerSequence`

Required – No

Default – JoinSequence

HorizontalLayoutConfiguration.TileCount

Description – Specifies the number of tiles that remain visible during a screen share

Type – Integer

Valid values – 1–10

Required – No

Default – 4

HorizontalLayoutConfiguration.TileAspectRatio

Description – Specifies the tiles' aspect ratio

Type – Integer

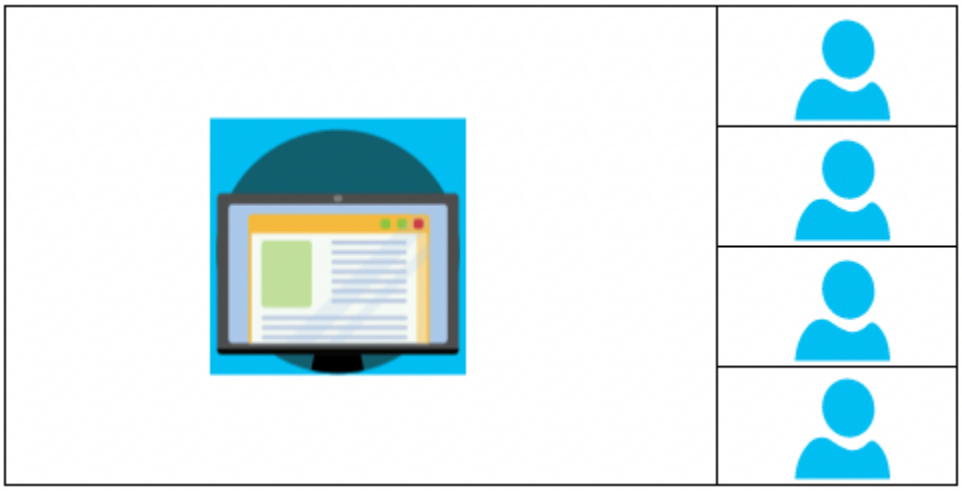
Valid values – *n***/***n*

Required – No

Default – 16/9, value applies to all tiles

VerticalLayoutConfiguration

`VerticalLayoutConfiguration` displays the content share and the four most recent videos stacked on the right. Presenters always appear on top. Other attendees appear in the order dictated by `TileOrder`.



The following example shows how to implement the vertical layout programmatically. In this case, the layout orders tiles by JoinSequence and places them to the right of the screen share. The layout allows a maximum of four tiles and applies a 16/9 aspect ratio.

```
{
   "CompositedVideo":{
      "Layout": "GridView",
      "Resolution": "FHD",
      "GridViewConfiguration":{
         "ContentShareLayout": "Vertical",
         "VerticalLayoutConfiguration":{
            "TileOrder": "JoinSequence",
            "TilePosition": "Right",
            "TileCount": 4,
            "TileAspectRatio": "16/9"
         }
      }
   }
}
```

VerticalLayoutConfiguration

Description – The configuration settings for a vertical layout

Type – VerticalLayoutConfiguration object

Required – No

VerticalLayoutConfiguration.TilePosition

Description – Places tiles to the right or left of a content share.

Type – String

Valid values – `Bottom` | `Top`

Required – No

Default – Bottom

VerticalLayoutConfiguration.TileOrder

Description – Orders tiles by when users join or when they speak

Type – String

Valid values – `JoinSequence` | `SpeakerSequence`

Required – No

Default – JoinSequence

VerticalLayoutConfiguration.TileCount

Description – Specifies the number of tiles

Type – Integer

Valid values – 1–10

Required – No

Default – 4

VerticalLayoutConfiguration.TileAspectRatio

Description – Specifies the tiles' aspect ratio

Type – Integer

Valid values – *n*/*n*

Required – No

Default – 9/16, value applies to all tiles

Creating media stream pipelines

Media stream pipelines capture individual audio for all the attendees in a meeting, plus the mixed audio generated by a media concatenation pipeline. All media stream pipelines save their data to Amazon Kinesis Video Streams (KVS).

You create the video stream by calling the CreateMediaPipelineKinesisVideoStreamPool API. You can create one media stream pipeline per Amazon Chime SDK meeting.

> **Note**
>
> If a meeting uses an opt-in Region as its MediaRegion, the KVS stream must be in that same Region. For example, if a meeting uses the `af-south-1` Region, the KVS stream must also be in `af-south-1`. However, if the meeting uses a Region that AWS turns on by default, the KVS stream can be in any available Region, including an opt-in Region. For example, if the meeting uses `ca-central-1`, the KVS stream can be in `eu-west-2`, `us-east-1`, `af-south-1`, or any other Region that the Amazon Chime SDK supports.
> To learn which AWS Region a meeting uses, call the GetMeeting API and use the MediaRegion parameter from the response.
> For more information about opt-in Regions, refer to Available AWS Regions for the Amazon Chime SDK service in this guide, and Specify which AWS Regions your account can use, in the *AWS Account Management Reference Guide*.

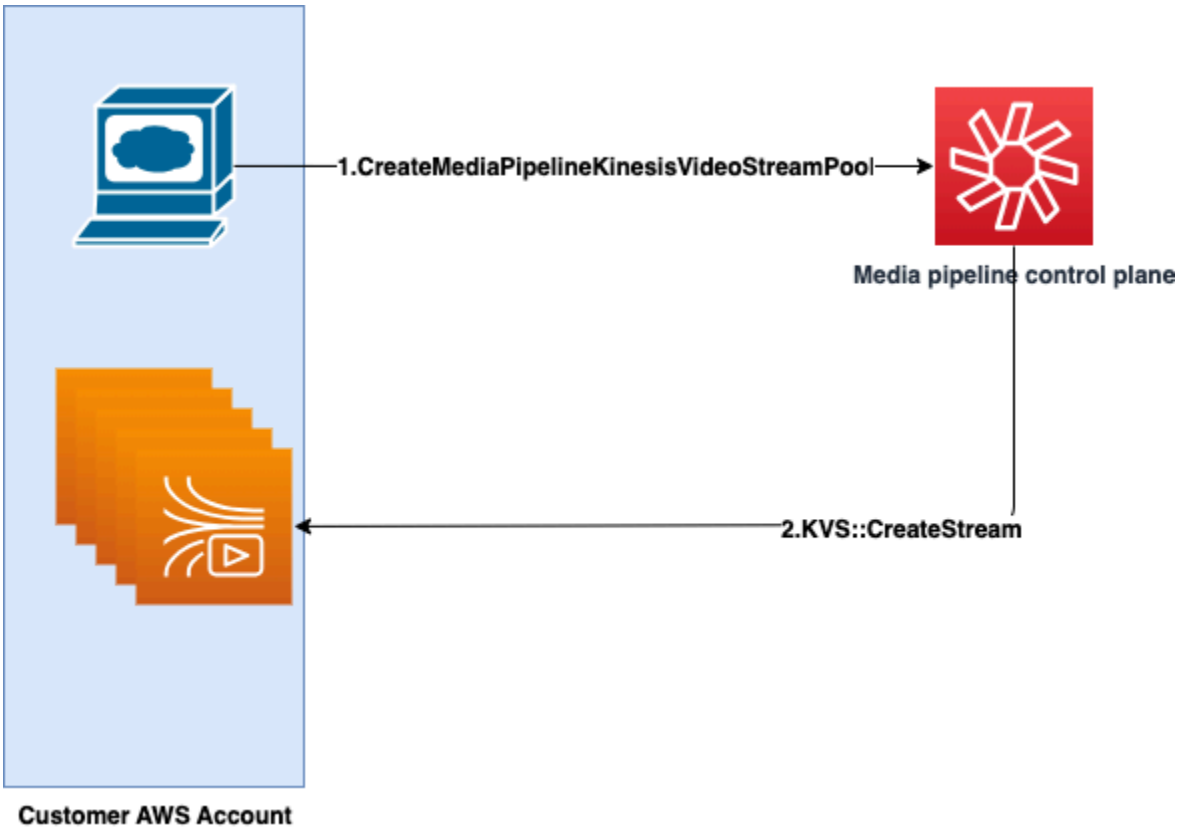
The following sections explain how to create a media stream pipeline. Follow them in the order listed.

Topics

Creating a Kinesis Video Streams pool

The Kinesis Video Streams (KVS) pool for your media stream pipeline must belong to the same AWS account as the Amazon Chime SDK meeting. You create a Kinesis Video Streams pool by calling the CreateMediaPipelineKinesisVideoStreamPool API.

The following diagram shows the architecture of a media pipeline Kinesis Video Streams pool. Numbers in the image correspond to the numbered text below:



In the diagram:

1. You call the CreateMediaPipelineKinesisVideoStreamPool API.
2. The media pipeline control plane creates and manages the Kinesis Video Streams (KVS) and the pool on your behalf in your account.

KVS pool operation, the process of creating, updating, and deleting streams in the pool, is asynchronous. As a result, Event Bridge notifications use the `Chime Media Pipeline Kinesis Video Pool State Change` detail type to communicate the status of the streams in a pool.

You can create a pool once and reuse it across different meetings. You can also create different pools as needed, and delete pools when you don't need them.

Pools scale up automatically, based on your concurrent call burst. You can delete any unneeded pools.

> **Note**
>
> When you delete a pool, you must wait for pool to be completely deleted before you delete the KVS streams in the pool. An Event Bridge notification will indicate when the pool has been completely deleted. That happens after all the meetings that use the pool have ended. You can also call the GetMediaPipelineKinesisVideoStreamPool API to view the `PoolId` for a given KVS pool.
> When you invoke the Kinesis Video Streams DeleteStream API, you can use that naming string to search for and delete the streams in a pool. You can also call the GetMediaPipelineKinesisVideoStreamPool API to view the `PoolId` for a given KVS pool. The examples in the next section explain how.

Example code for Kinesis Video Streams pools

The following examples show how to create, update, get, list, and delete Kinesis Video Streams (KVS) pools. Expand each section to learn more.

Imports and common variables

```
'''
Define imports and common variables
'''

import boto3
from uuid import uuid4
import json

client = boto3.client("chime-sdk-media-pipelines", region_name='us-east-1')
pool_name = 'MyDemoKvsPool'

def pretty_print_json(obj):
    print(json.dumps(obj, default=str, indent=4))
```

CreateMediaPipelineKinesisVideoStreamPool

```
response = client.create_media_pipeline_kinesis_video_stream_pool(
        StreamConfiguration={
            'Region': 'us-east-1',
            'DataRetentionInHours': 24
        },
        PoolName=pool_name,
        ClientRequestToken=str(uuid4()),
        Tags=[
            {
                'Key': 'MyTagForAccessControl',
                'Value': 'SomeTagValue'
            },
        ]
)

pretty_print_json(response['KinesisVideoStreamPoolConfiguration'])
```

Output:

```
{
    "PoolArn": "arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-video-stream-
pool/MyDemoKvsPool",
    "PoolName": "MyDemoKvsPool",
    "PoolId": "ChimeMediaPipelines-MyDemoKvsPool-1f4e1a69-e718-4884-bf92-8a393ac0405b",
    "PoolStatus": "CREATING",
    "StreamConfiguration": {
        "Region": "us-east-1",
        "DataRetentionInHours": 24
    },
    "CreatedTimestamp": "2023-10-13 01:26:09.979000+00:00",
    "UpdatedTimestamp": "2023-10-13 01:26:09.979000+00:00"
}
```

GetMediaPipelineKinesisVideoStream

```
response = client.get_media_pipeline_kinesis_video_stream_pool(
        Identifier=pool_name
)

pretty_print_json(response['KinesisVideoStreamPoolConfiguration'])
```

Output:

```
{
    "PoolArn": "arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-video-stream-pool/MyDemoKvsPool",
    "PoolName": "MyDemoKvsPool",
    "PoolId": "ChimeMediaPipelines-MyDemoKvsPool-1f4e1a69-e718-4884-bf92-8a393ac0405b",
    "PoolStatus": "ACTIVE",
    "StreamConfiguration": {
        "Region": "us-east-1",
        "DataRetentionInHours": 24
    },
    "CreatedTimestamp": "2023-10-13 01:26:09.979000+00:00",
    "UpdatedTimestamp": "2023-10-13 01:26:09.979000+00:00"
}
```

UpdateMediaPipelineKinesisVideoStream

```
response = client.update_media_pipeline_kinesis_video_stream_pool(
    Identifier=pool_name,
    StreamConfiguration={
        'DataRetentionInHours': 48
    }
)
pretty_print_json(response['KinesisVideoStreamPoolConfiguration'])
```

Output:

```
{
    "PoolArn": "arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-video-stream-pool/MyDemoKvsPool",
    "PoolName": "MyDemoKvsPool",
    "PoolId": "ChimeMediaPipelines-MyDemoKvsPool-d08c26ae-0336-4e2e-acdf-805a7d71b891",
    "PoolStatus": "UPDATING",
    "PoolSize": 40,
    "StreamConfiguration": {
        "Region": "us-east-1",
        "DataRetentionInHours": 48
    },
    "CreatedTimestamp": "2023-10-13 01:44:23.010000+00:00",
    "UpdatedTimestamp": "2023-10-13 01:44:28.486000+00:00"
}
```

ListMediaPipelineKinesisVideoStream

```
list_of_pools = []
max_results = 100
next_token = None
while(True):
    if next_token:
        response = client.list_media_pipeline_kinesis_video_stream_pools(
            NextToken=next_token,
            MaxResults=max_results
        )
    else:
        response = client.list_media_pipeline_kinesis_video_stream_pools(
            MaxResults=max_results
        )

    list_of_pools.extend(response['KinesisVideoStreamPools'])
    next_token = response.get('NextToken')
    if not next_token:
        break
pretty_print_json(list_of_pools)
```

Output:

```
[
    {
        "PoolName": "MyDemoKvsPool",
        "PoolId": "ChimeMediaPipelines-MyDemoKvsPool-6588e703-f046-4288-
ba7f-0c03de76a6bb",
        "PoolArn": "arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-video-
stream-pool/MyDemoKvsPool"
    }
]
```

DeleteMediaPipelineKinesisVideoStream

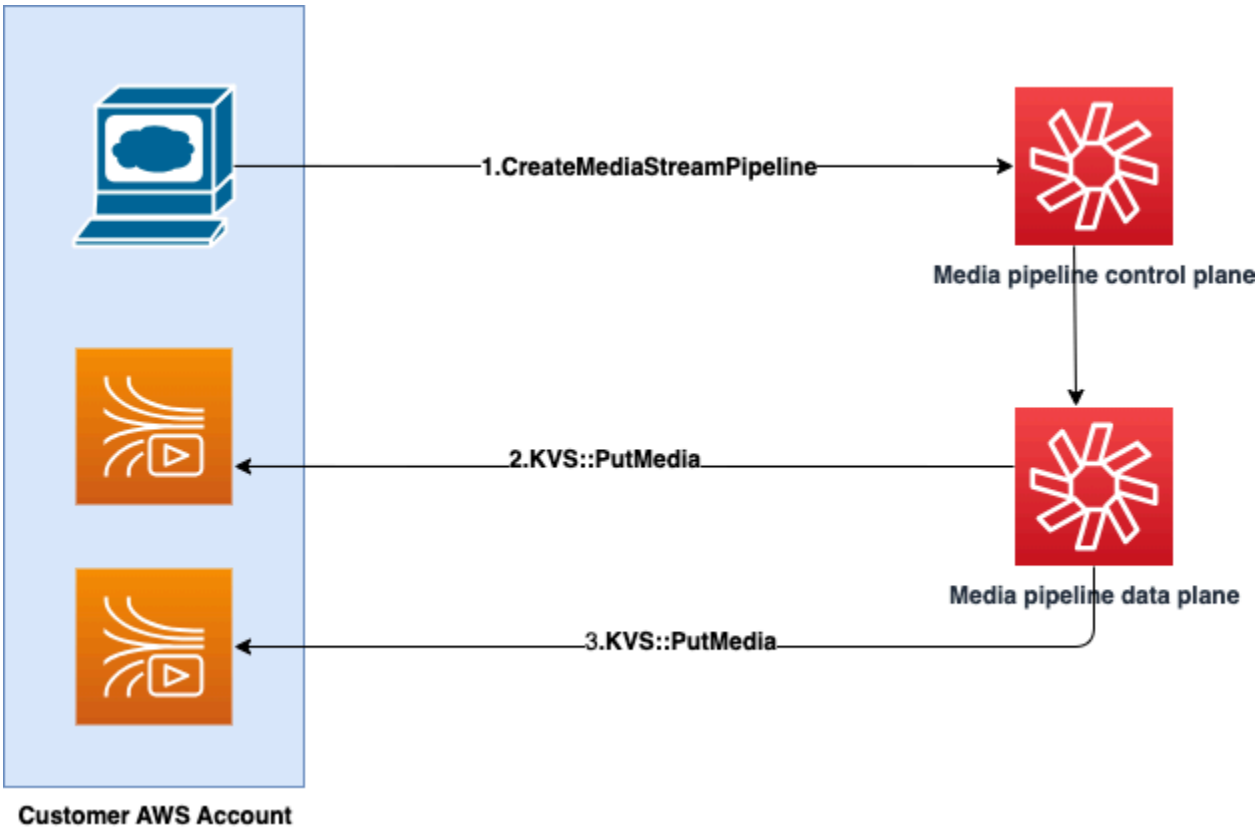
```
client.delete_media_pipeline_kinesis_video_stream_pool(
    Identifier=pool_name
)
```

Output: A successful `delete_media_pipeline_kinesis_video_stream_pool` request has no body.

Creating media stream pipelines

The chime media stream pipeline must belong to the same AWS account as the Amazon Chime SDK meeting. You create the Amazon Chime SDK media stream pipeline by calling the CreateMediaStreamPipeline API and specifying a source and a sink.

The following diagram shows the architecture of an Amazon Chime SDK media stream pipeline. Numbers in the diagram correspond to the numbered text below.



In the diagram:

1. You call the `CreateMediaStreamPipeline` API. In the request, specify the sources and sinks for the streams. whether you want to capture individual audio, mixed audio, or both. Include the ARN of your KVS pool in the request.
   - The sources array consists of the `SourceType` and `SourceArn`. You must use the `ChimeSdkMeeting SourceType`. The `SourceArn` is the ARN of the `ChimeSdkMeeting`.
   - The sinks array consists of the `SinkType`, `SinkArn`, `ReservedStreamCapacity`, and `MediaStreamType`. We only support the `KinesisVideoStreamPoolSinkType`. The `SinkArn` is the ARN of the `KinesisVideoStreamPool`. The `MediaStreamType` controls the type of media streamed to the sink, either `MixedAudio` or `IndividualAudio`.

`ReservedStreamCapacity` sets the number of streams allocated for the `MediaStreamType` from the `KinesisVideoStreamPool`.

- If you want to stream both `IndividualAudio` and `MixedAudio`, create two sink objects in the `Sinks` array, one for `IndividualAudio`, another for `MixedAudio`. The `SinkArn` (the ARN of the `KinesisVideoStreamPool`) can vary for each sink.
- To stream just individual audio or mixed audio, create one sink object with the desired `MediaStreamType`.

- Note the following:
  - When invoking the CreateMediaStreamPipeline API with `KinesisVideoStreamPool` as the `SinkType`, the `SinkARN` must belong to the control plane region in which `CreateMediaStreamPipeline` is being invoked.

    For example, if you create a media stream pipeline in `us-east-1`, you must use a `KinesisVideoStreamPool` in `us-east-1`.
  - `ReservedStreamCapacity` should be **1** when you specify the `MixedAudio` `MediaStreamType`, and between **1-10** when you specify the `IndividualAudio` `MediaStreamType`.

2. The media pipeline data plane calls the KVS PutMedia API to store individual audio in a KVS stream that belongs to the KVS pool that you specify.
3. The media pipeline data plane calls the KVS `PutMedia` API to store mixed audio in a stream that belongs to the KVS pool that you specify.

> **Note**
>
> After calling the CreateMediaStreamPipeline API, builders can use media pipeline events or call the GetMediaPipeline API to determine if the pipeline state is `InProgress`.
> Once the pipeline state reaches `InProgress`, the media—any combination of `IndividualAudio` and `MixedAudio`—streams to KVS.
> For the `IndividualAudio` stream type, a 1:1 mapping exists between attendee IDs and the KVS stream allocated from the `KinesisVideoStreamPool`. The mapping applies for the life of the media pipeline.
> To know which KVS stream maps to an attendee ID, or is assigned for MixedAudio, use one of the following techniques:

- Use Event Bridge Notifications. Each notification provides information such as attendee IDs and the KVS ARN that streams the attendee's audio. When a `IndividualAudio` or `MixedAudio` streaming session starts, we send a `chime:MediaPipelineKinesisVideoStreamStart` event. Streaming sessions end when an attendee leaves the call (for `IndividualAudio`), or when the meeting ends.
- Use the persistent metadata that the Kinesis Video Streams send with each fragment. The metadata contains information similar to what Event Bridge sends. Builders need to parse all the streams of the `KinesisVideoStreamPool` by specifying the pool name as the prefix in the ListStreams Kinesis Video Streams API using this solution.

Media Stream pipeline termination happens when the meeting is deleted, or the DeleteMediaPipeline API is invoked for that media stream pipeline. An Event Bridge notification is also sent to indicate the media pipeline termination.

Example code for media stream pipelines

The following examples show how to create media stream pipelines for mixed audio, individual audio, and both. Expand each section to learn more.

CreateMediaStreamPipeline for mixed audio

```
response = client.create_media_stream_pipeline(
    Sources=[
        {
            'SourceType': 'ChimeSdkMeeting',
            'SourceArn': 'arn:aws:chime:us-east-1:account-
ID:meeting/bed804cf-8cf0-4991-9b8d-d1acc2987433'
        },
    ],
    Sinks=[
        {
            'SinkArn': 'arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-
video-stream-pool/foo',
            'SinkType': 'KinesisVideoStreamPool',
            'ReservedStreamCapacity': 1,
            'MediaStreamType': 'MixedAudio'
        },
    ],
```

```
    ClientRequestToken='sample token',
    Tags=[
        {
            'Key': 'sample key',
            'Value': 'sample value'
        },
    ]
)
```

Response:

```
{
    'MediaStreamPipeline': {
        'MediaPipelineId': '45bc79a0-4591-4ebe-a642-d42c4e279f2d',
        'MediaPipelineArn': 'arn:aws:chime:us-east-1:account-ID:media-
pipeline/45bc79a0-4591-4ebe-a642-d42c4e279f2d',
        'CreatedTimestamp': '2023-07-25T21:48:48.265Z',
        'UpdatedTimestamp': '2023-07-25T21:48:48.376Z',
        'Status': 'Initializing',
        'Sources': [
            {
                'SourceType': 'ChimeSdkMeeting',
                'SourceArn': 'arn:aws:chime:us-east-1:account-
ID:meeting/bed804cf-8cf0-4991-9b8d-d1acc2987433'
            },
        ],
        'Sinks': [
            {
                'SinkArn': 'arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-
video-stream-pool/foo',
                'SinkType': 'KinesisVideoStreamPool',
                'ReservedStreamCapacity': 1,
                'MediaStreamType': 'MixedAudio'
            },
        ]
    }
}
```

CreateMediaStreamPipeline for individual audio

```
response = client.create_media_stream_pipeline(
    Sources=[
        {
```

```
            'SourceType': 'ChimeSdkMeeting',
            'SourceArn': 'arn:aws:chime:us-east-1:account-
ID:meeting/bed804cf-8cf0-4991-9b8d-d1acc2987433'
        },
    ],
    Sinks=[
        {
            'SinkArn': 'arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-
video-stream-pool/foo',
            'SinkType': 'KinesisVideoStreamPool',
            'ReservedStreamCapacity': 5,
            'MediaStreamType': 'IndividualAudio'
        },
    ],
    ClientRequestToken='sample token',
    Tags=[
        {
            'Key': 'sample key',
            'Value': 'sample value'
        },
    ]
)
```

Response:

```
{
    'MediaStreamPipeline': {
        'MediaPipelineId': '45bc79a0-4591-4ebe-a642-d42c4e279f2d',
        'MediaPipelineArn': 'arn:aws:chime:us-east-1:account-ID:media-
pipeline/45bc79a0-4591-4ebe-a642-d42c4e279f2d',
        'CreatedTimestamp': '2023-07-25T21:48:48.265Z',
        'UpdatedTimestamp': '2023-07-25T21:48:48.376Z',
        'Status': 'Initializing',
        'Sources': [
            {
                'SourceType': 'ChimeSdkMeeting',
                'SourceArn': 'arn:aws:chime:us-east-1:account-
ID:meeting/bed804cf-8cf0-4991-9b8d-d1acc2987433'
            },
        ],
        'Sinks': [
            {
```

```
                'SinkArn': 'arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-
video-stream-pool/foo',
                'SinkType': 'KinesisVideoStreamPool',
                'ReservedStreamCapacity': 5,
                'MediaStreamType': 'IndividualAudio'
            },
        ]
    }
}
```

CreateMediaStreamPipeline for mixed and individual audio

```
response = client.create_media_stream_pipeline(
    Sources=[
        {
            'SourceType': 'ChimeSdkMeeting',
            'SourceArn': 'arn:aws:chime:us-east-1:account-
ID:meeting/bed804cf-8cf0-4991-9b8d-d1acc2987433'
        },
    ],
    Sinks=[
        {
            'SinkArn': 'arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-
video-stream-pool/foo',
            'SinkType': 'KinesisVideoStreamPool',
            'ReservedStreamCapacity': 1,
            'MediaStreamType': 'MixedAudio'
        },
        {
            'SinkArn': 'arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-
video-stream-pool/foo',
            'SinkType': 'KinesisVideoStreamPool',
            'ReservedStreamCapacity': 5,
            'MediaStreamType': 'IndividualAudio'
        },
    ],
    ClientRequestToken='sample token',
    Tags=[
        {
            'Key': 'sample key',
            'Value': 'sample value'
        },
    ]
```

```
)
```

Response:

```
{
    'MediaStreamPipeline': {
        'MediaPipelineId': '45bc79a0-4591-4ebe-a642-d42c4e279f2d',
        'MediaPipelineArn': 'arn:aws:chime:us-east-1:account-ID:media-
pipeline/45bc79a0-4591-4ebe-a642-d42c4e279f2d',
        'CreatedTimestamp': '2023-07-25T21:48:48.265Z',
        'UpdatedTimestamp': '2023-07-25T21:48:48.376Z',
        'Status': 'Initializing',
        'Sources': [
            {
                'SourceType': 'ChimeSdkMeeting',
                'SourceArn': 'arn:aws:chime:us-east-1:account-
ID:meeting/bed804cf-8cf0-4991-9b8d-d1acc2987433'
            },
        ],
        'Sinks': [
            {
                'SinkArn': 'arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-
video-stream-pool/foo',
                'SinkType': 'KinesisVideoStreamPool',
                'ReservedStreamCapacity': 1,
                'MediaStreamType': 'MixedAudio'
            },
            {
                'SinkArn': 'arn:aws:chime:us-east-1:account-ID:media-pipeline-kinesis-
video-stream-pool/foo',
                'SinkType': 'KinesisVideoStreamPool',
                'ReservedStreamCapacity': 5,
                'MediaStreamType': 'IndividualAudio'
            },
        ]
    }
}
```

Using Event Bridge notifications

In addition to the Using media pipeline events, media stream pipelines send Event Bridge notifications when they start and stop streaming to KVS, and when video pool states change.

Topics

- Understanding media stream pipeline events
- Understanding media pipeline Kinesis Video Stream pool events

Understanding media stream pipeline events

Media stream pipelines send the following events. Expand each section to learn more.

Amazon Chime Media Stream Pipeline Kinesis Video Stream Start

The Amazon Chime SDK media pipeline sends this event when the media stream pipeline starts receiving audio from the meeting and streaming that audio to KVS. Empty `AttendeeId` and `ExternalUserId` fields indicate that the media pipeline sent mixed audio to the KVS stream.

```
{
    "version": "0",
    "id": "5ee6265a-0a40-104e-d8fd-a3b4bdd78483",
    "detail-type": "Chime Media Pipeline State Change",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "2021-07-28T20:20:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventType": "chime:MediaPipelineKinesisVideoStreamStart",
        "timestamp": 1627503649251,
        "meetingId": "1e6bf4f5-f4b5-4917-b8c9-bda45c340706",
        "externalMeetingId": "Meeting_Id",
        "mediaPipelineId": "e40ee45e-2ed1-408e-9156-f52b8208a491",
        "mediaRegion": "ap-southeast-1",

        "attendeeId": "Attendee_Id",
        "externalUserId": "External_User_Id",

        "kinesisVideoStreamArn": "arn:aws:kinesisvideo:us-east-1:123456:stream/Chime*",
        "startFragmentNumber": "1234567899444",
        "startTime": "yyyy-mm-ddThh:mm:ssZ"

    }
}
```

Amazon Chime Media Stream Pipeline Kinesis Video Stream End

The media pipeline sends this event to Event Bridge when streaming to KVS ends.

```
{
    "version": "0",
    "id": "5ee6265a-0a40-104e-d8fd-a3b4bdd78483",
    "detail-type": "Chime Media Pipeline State Change",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "2021-07-28T20:20:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventType": "chime:MediaPipelineKinesisVideoStreamEnd",
        "timestamp": 1627503649251,
        "meetingId": "1e6bf4f5-f4b5-4917-b8c9-bda45c340706",
        "externalMeetingId": "Meeting_Id",
        "mediaPipelineId": "e40ee45e-2ed1-408e-9156-f52b8208a491",
        "mediaRegion": "ap-southeast-1",

        "attendeeId": "Attendee_Id",
        "externalUserId": "External_User_Id",

        "kinesisVideoStreamArn": "arn:aws:kinesisvideo:us-east-1:123456:stream/Chime*",
        "startFragmentNumber": "1234567899444",
        "startTime": "yyyy-mm-ddThh:mm:ssZ",
        "endTime": "yyyy-mm-ddThh:mm:ssZ",
        "endFragmentNumber": "1234567899555"
    }
}
```

Understanding media pipeline Kinesis Video Stream pool events

Media pipelines send the following events to Event Bridge when the pools' states change. Expand each section to learn more.

Amazon Chime Media Pipeline Kinesis Video Pool Active

The media pipeline sends this event is sent after the CreateMediaPipelineKinesisVideoStreamPool API creates a pool.

```
{
```

```
    "version": "0",
    "id": "5ee6265a-0a40-104e-d8fd-a3b4bdd78483",
    "detail-type": "Chime Media Pipeline Kinesis Video Pool State Change",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "2021-07-28T20:20:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventType": "chime:MediaPipelineKinesisVideoStreamPoolActive",
        "timestamp": 1627503649251,
        "mediaRegion": "ap-southeast-1",
        "poolArn" : "ARN of the KVS Pool"
    }
}
```

Amazon Chime Chime Media Pipeline Kinesis Video Pool Updated

The media pipeline sends this event after the UpdateMediaPipelineKinesisVideoStreamPool API updates a pool.

```
{
    "version": "0",
    "id": "5ee6265a-0a40-104e-d8fd-a3b4bdd78483",
    "detail-type": "Chime Media Pipeline Kinesis Video Pool State Change",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "2021-07-28T20:20:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventType": "chime:MediaPipelineKinesisVideoStreamPoolUpdated",
        "timestamp": 1627503649251,
        "mediaRegion": "ap-southeast-1",
        "poolArn" : "ARN of the KVS Pool"
    }
}
```

Amazon Chime Media Pipeline Kinesis Video Pool Deleted

The media pipeline sends this event to Event Bridge when the DeleteMediaPipelineKinesisVideoStreamPool deletes a pool.

For more information about deleting pools, refer to Creating a Kinesis Video Streams pool, in this section.

```
{
    "version": "0",
    "id": "5ee6265a-0a40-104e-d8fd-a3b4bdd78483",
    "detail-type": "Chime Media Pipeline Kinesis Video Pool State Change",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "2021-07-28T20:20:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {If the attendeeId and externalUserId fields are empty, the media
 pipeline sends mixed audio to the KVS stream.
        "eventType": "chime:MediaPipelineKinesisVideoStreamPoolDeleted",
        "timestamp": 1627503649251,
        "mediaRegion": "ap-southeast-1",
        "poolArn" : "ARN of the KVS Pool"
    }
 }
```

Amazon Chime Media Pipeline Kinesis Video Pool Temporary Failure

The media pipeline sends the following event to Event Bridge when a video pool fails temporarily.

```
{
    "version": "0",
    "id": "5ee6265a-0a40-104e-d8fd-a3b4bdd78483",
    "detail-type": "Chime Media Pipeline Kinesis Video Pool State Change",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "2021-07-28T20:20:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventType": "chime:MediaPipelineKinesisVideoStreamPoolTemporaryFailure",
        "timestamp": 1627503649251,
        "mediaRegion": "ap-southeast-1",
        "poolArn" : "ARN of the KVS Pool"
    }
 }
```

Amazon Chime Media Pipeline Kinesis Video Pool Permanent Failure

The media pipeline sends the following event to Event Bridge when a video pool fails permanently.

```
{
    "version": "0",
    "id": "5ee6265a-0a40-104e-d8fd-a3b4bdd78483",
    "detail-type": "Chime Media Pipeline Kinesis Video Pool State Change",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "2021-07-28T20:20:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventType": "chime:MediaPipelineKinesisVideoStreamPoolPermanentFailure",
        "timestamp": 1627503649251,
        "mediaRegion": "ap-southeast-1",
        "poolArn" : "ARN of the KVS Pool"
    }
}
```

Using media stream pipeline data

You can use the metadata in the notifications to get KVS ARNs, fragment numbers, and fragment timestamps. That information can help you to process the audio data in a KVS stream.

Also, you can use KVS ARNs with the KVS APIs to read data from a stream. Depending on the use case, you call the GetMedia and GetMediaForFragmentList APIs. Typically, a `GetMediaForFragmentList` call is preceded by a call to the ListFragments API. For more information, refer Reading data from streams, in the *Amazon Kinesis Video Streams FAQs*.

Depending on the use case, builders can use the Kinesis Video Streams parser library, which in turn uses the KVS GetMedia API.

Media stream pipelines add the following meeting and attendee metadata to each fragment.

```
"meetingId"
"externalMeetingId"
"attendeeId"
"externalUserId"
"sampleRate"
"channels"
```

Media data is stored in MKV format. All MKV audio data is AAC encoded. For more information, see Kinesis Video Streams data model, in the *Kinesis Video Streams Developer Guide*.

Creating a service-linked role for media pipelines

The information in the following sections explains how to create a service-linked role that grants media pipelines access to your Amazon Chime SDK meetings.

Topics

Setting role permissions

media pipelines use a service-linked role named *AWSServiceRoleForAmazonChimeSDKMediaPipelines*. The role allows the capture pipelines to access Amazon Chime SDK meetings and publish metrics to Amazon CloudWatch on your behalf. The role trusts the `mediapipelines.chime.amazonaws.com` service.

The role permissions policy allows the Amazon Chime SDK to complete the following actions on all AWS resources:

- Action: `cloudwatch:PutMetricData` on `all AWS resources`
- Action: `chime:CreateAttendee` on `all AWS resources`
- Action: `chime:DeleteAttendee` on `all AWS resources`
- Action: `chime:GetMeeting` on `all AWS resources`
- Action: `kinesisvideo:CreateStream` on `arn:aws:kinesisvideo:*:`*`111122223333`*`:stream/ChimeMediaPipelines-*`
- Action: `kinesisvideo:PutMedia` on `arn:aws:kinesisvideo:*:`*`111122223333`*`:stream/ChimeMediaPipelines-*`
- Action: `kinesisvideo:UpdateDataRetention` on `arn:aws:kinesisvideo:*:`*`111122223333`*`:stream/ChimeMediaPipelines-*`

- Action: `kinesisvideo:DescribeStream` on `arn:aws:kinesisvideo:*:111122223333:stream/ChimeMediaPipelines-*`
- Action: `kinesisvideo:GetDataEndpoint` on `arn:aws:kinesisvideo:*:111122223333:stream/ChimeMediaPipelines-*`
- Action: `kinesisvideo:ListStreams` on `arn:aws:kinesisvideo:*:111122223333:stream/*`

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 about permissions, see Service linked role permissions in the *IAM User Guide*.

Creating the service-linked role

You use the IAM console to create a service-linked role for use with Amazon Chime SDK media pipelines. You must have IAM administrative permissions to complete these steps. If you don't, contact a system administrator.

To create the role

1. Sign in to the AWS Management Console, and then open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
3. Choose the **AWS Service** role type, and then choose **Chime SDK Media Pipelines**.

   The IAM policy appears.
4. Select the check box next to the policy, then choose **Next: Tags**.
5. Choose **Next: Review**.
6. Edit the description as needed, then choose **Create role**.

You can also use the AWS CLI or the AWS API to create a service-linked role named *mediapipelines.chime.amazonaws.com*. In the AWS CLI, run this command:

```
aws iam create-service-linked-role --aws-service-name
 mediapipelines.chime.amazonaws.com
```

For more information creating the role, see Creating a Service-Linked Role in the *IAM User Guide*. If you delete this role, you can use this same process to create it again.

Editing the service-linked role

You can't to edit the *AWSServiceRoleForAmazonChimeSDKMediaPipelines* service-linked role. After you create the role, you can't change its name because other entities may reference the role. However, you can use IAM to edit the role's description. For more information, see Editing a Service-Linked Role in the *IAM User Guide*.

Deleting the service-linked role

If don't need a service-linked role, we recommend that you delete it. To do that, you first delete the media pipelines that use the role. You can use the AWS CLI or the DeleteMediaCapturePipeline API to delete the pipelines.

Using the CLI to delete pipelines

Use this command in the AWS CLI to delete media pipelines in your account.

```
aws chime-sdk-media-pipelines delete-media-capture-pipeline --media-pipeline-
id Pipeline_Id
```

Using an API to delete pipelines

Use the DeleteMediaCapturePipeline API to delete media pipelines in your account.

Deleting the role

Once you delete the pipelines, you can use the IAM console, the AWS CLI, or the AWS API to delete the role. For more information about deleting roles, see Deleting a Service-Linked Role in the *IAM User Guide*.

Regions that support service-linked roles

Amazon Chime SDK supports using service-linked roles in all of the AWS Regions where the service is available. For more information, see Amazon Chime SDK endpoints and quotas in the *Amazon Web Services General Reference*.

Using media pipeline events

Each type of media pipeline sends lifecycle events, which you can use to trigger notifications and initiate downstream workflows. Some examples of using media pipeline events include:

- Processing captured media after a media pipeline has completed.

- Notifying meeting participants if a media pipeline has a temporary failure.
- Stopping a meeting if a media pipeline fails permanently.

You can send events to Amazon EventBridge, Amazon Simple Notification Service (SNS), and Amazon Simple Queue Service (SQS). For more information, refer to Events from AWS services in the *Amazon EventBridge User Guide*.

Amazon Chime SDK media pipeline created

The Amazon Chime SDK sends this event when the media pipeline is created.

Example: Event data

The following is example data for this event.

```
{
    "version": "0",
    "id": "5ee6265a-0a40-104e-d8fd-a3b4bdd78483",
    "detail-type": "Chime Media Pipeline State Change",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "2021-07-28T20:20:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "version": "0",
        "eventType": "chime:MediaPipelineInProgress",
        "timestamp": 1627503649251,
        "meetingId": "1e6bf4f5-f4b5-4917-b8c9-bda45c340706",
        "externalMeetingId": "Meeting_Id",
        "mediaPipelineId": "e40ee45e-2ed1-408e-9156-f52b8208a491",
        "mediaRegion": "ap-southeast-1"
    }
}
```

Amazon Chime SDK media pipeline deleted

The Amazon Chime SDK sends this event after the media pipeline stops successfully.

Example: Event data

The following is example data for this event.

```
{
    "version": "0",
    "id": "9e11e429-97fd-9532-5670-fac3f7abc05f",
    "detail-type": "Chime Media Pipeline State Change",
    "source": "aws.chime",
    "account": "365135496707",
    "time": "2021-07-28T20:21:50Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "version": "0",
        "eventType": "chime:MediaPipelineDeleted",
        "timestamp": 1627503710485,
        "meetingId": "1e6bf4f5-f4b5-4917-b8c9-bda45c340706",
        "externalMeetingId": "Meeting_Id",
        "mediaPipelineId": "e40ee45e-2ed1-408e-9156-f52b8208a491",
        "mediaRegion": "ap-southeast-1"
    }
}
```

Amazon Chime SDK media pipeline has a temporary failure

The Amazon Chime SDK sends this event when the media pipeline has a temporary failure.

Example: Event data

The following is example data for this event.

```
{
    "version": "0",
    "id": "abc141e1-fc2e-65e8-5f18-ab5130f1035a",
    "detail-type": "Chime Media Pipeline State Change",
    "source": "aws.chime",
    "account": "365135496707",
    "time": "2021-07-28T21:16:42Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "version": "0",
        "eventType": "chime:MediaPipelineTemporaryFailure",
        "timestamp": 1627507002882,
        "meetingId": "7a5434e3-724a-4bbb-9eb6-2fb209dc0706",
        "externalMeetingId": "Meeting_Id",
```

```
        "mediaPipelineId": "ebd62f4e-04a9-426d-bcb0-974c0f266400",
        "mediaRegion": "eu-south-1"
    }
}
```

Amazon Chime SDK media pipeline in progress

The Amazon Chime SDK sends this event when the media pipeline begins capturing artifacts.

Example: Event data

The following is example data for this event.

```
{
    "version": "0",
    "id": "9e11e429-97fd-9532-5670-fac3f7abc05f",
    "detail-type": "Chime Media Pipeline State Change",
    "source": "aws.chime",
    "account": "365135496707",
    "time": "2021-07-28T20:21:50Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "version": "0",
        "eventType": "chime:MediaPipelineInProgress",
        "timestamp": 1627503710485?,
        "meetingId": "1e6bf4f5-f4b5-4917-b8c9-bda45c340706",
        "externalMeetingId": "Meeting_Id",
        "mediaPipelineId": "e40ee45e-2ed1-408e-9156-f52b8208a491",
        "mediaRegion": "ap-southeast-1"
    }
}
```

Amazon Chime SDK media pipeline permanent failure

The Amazon Chime SDK sends this event when a media pipeline fails permanently.

Example: Event data

The following is example data for this event.

```
{
    "version": "0",
    "id": "9e11e429-97fd-9532-5670-fac3f7abc05f",
```

```
    "detail-type": "Chime Media Pipeline State Change",
    "source": "aws.chime",
    "account": "365135496707",
    "time": "2021-07-28T20:21:50Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "version": "0",
        "eventType": "chime:MediaPipelinePermanentFailure",
        "timestamp": 1627503710485,
        "meetingId": "1e6bf4f5-f4b5-4917-b8c9-bda45c340706",
        "externalMeetingId": "Meeting_Id",
        "mediaPipelineId": "e40ee45e-2ed1-408e-9156-f52b8208a491",
        "mediaRegion": "ap-southeast-1"
    }
}
```

Setting Amazon S3 bucket permissions

If you haven't created an Amazon S3 bucket, make sure you create yours in the account and Region where you host meetings. Also, make sure you grant adequate permissions to the service. For more information about creating an Amazon S3 bucket, see Creating an Amazon S3 bucket.

Sending media pipeline events to CloudTrail

AWS enables CloudTrail for you when you create your AWS account. When a user calls a supported API in the media pipeline SDK, CloudTrail logs that activity for that API in **Event history**, along with other AWS events. You can view, search, and download the media pipeline events in your AWS account. For more information, see Viewing Events with CloudTrail Event History in the *CloudTrail User Guide*.

For an ongoing record of media pipeline events, you can create a *trail*. A trail enables CloudTrail to deliver log files to your Amazon S3 bucket. The following example shows a media pipeline trail. The data includes the user that called the API, the IAM role used to call the API, and timestamps. For more information about using CloudTrail see Logging and monitoring in the *Amazon Chime SDK Administrator Guide*.

```
{
   "Records": [
   {
      "eventVersion": "1.08",
      "userIdentity": {
```

```
            "type": "AssumedRole",
            "principalId": "ABCDEFGHIJKLMNOPQRSTUV:user-name",
            "arn": "arn:aws:sts::123456789101:assumed-role/role-name/user-name",
            "accountId": "109876543210",
            "accessKeyId": "ABCDEFGHIJKLMNOPQRSTUV",
            "sessionContext": {
                "sessionIssuer": {
                    "type": "Role",
                    "principalId": "ABCDEFGHIJKLMNOPQRSTUV",
                    "arn": "arn:aws:iam::109876543210:role/role-name",
                    "accountId": "012345678910",
                    "userName": "user-name"
                    },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2022-03-08T19:34:55Z"
                }
            }
        },
        "eventTime": "2022-03-08T20:28:41Z",
       "eventSource": "chime-sdk-media-pipelines.amazonaws.com",
       "eventName": "CreateMediaCapturePipeline",
       "awsRegion": "us-east-1",
       "sourceIPAddress": "127.0.0.1",
       "userAgent": "[]/[]",
       "requestParameters": {
           "sourceType": "ChimeSdkMeeting",
           "sourceArn": "Hidden_For_Security_Reasons",
           "sinkType": "S3Bucket",
           "sinkArn": "Hidden_For_Security_Reasons",
           "chimeSdkMeetingConfiguration": {
               "artifactsConfiguration": {
                   "audio": {
                      "muxType": "AudioOnly"
                   },
               "video": {
                   "state": "Enabled",
                   "muxType": "VideoOnly"
                   },
               "content": {
                   "state": "Enabled",
                   "muxType": "ContentOnly"
                   }
```

```
            }
        }
      },
     "responseElements": {
        "mediaCapturePipeline": {
        "mediaPipelineId": "pipeline-uuid",
        "sourceType": "ChimeSdkMeeting",
        "sourceArn": "Hidden_For_Security_Reasons",
        "status": "Initializing",
        "sinkType": "S3Bucket",
        "sinkArn": "Hidden_For_Security_Reasons",
        "createdTimestamp": "2022-03-08T20:28:41.336Z",
        "updatedTimestamp": "2022-03-08T20:28:41.463Z",
        "chimeSdkMeetingConfiguration": {
            "artifactsConfiguration": {
                "audio": {
                    "muxType": "AudioOnly"
                },
            "video": {
                "state": "Enabled",
                 "muxType": "VideoOnly"
                 },
             "content": {
                 "state": "Enabled",
                 "muxType": "ContentOnly"
                 }
              }
            }
          }
      },
      "requestID": "request-id",
      "eventID": "event-id",
     "readOnly": false,
      "eventType": "AwsApiCall",
      "managementEvent": true,
      "eventCategory": "Management",
      "recipientAccountId": "112233445566",
      "tlsDetails": {
          "tlsVersion": "TLSv1.2",
          "clientProvidedHostHeader": "example.com"
       }
    },
  ]
```

```
}
```

Parsing transcripts

Use the following command to parse transcription content from a transcription message. The command parses complete sentences from the transcript-message.txt files.

```
with open('transcript-message.txt') as f:
        for line in f:
            result_json = json.loads(line)["transcript"]["results"][0]
            if result_json['isPartial'] == False:
                print(result_json["alternatives"][0]["transcript"])
```

Best practices for stopping pipelines

As a best practice for stopping media pipelines, call the DeleteMediaPipeline API. The API allows you to delete media capture and media live connector pipelines. You can also call the DeleteMediaCapturePipeline API to delete media capture pipelines. All media pipelines stop when the meeting ends.

Using Amazon Chime SDK live transcription

You use Amazon Chime SDK live transcription to generate live, user-attributed transcripts of your meetings. Amazon Chime SDK live transcription integrates with the Amazon Transcribe and Amazon Transcribe Medical services to generate transcripts of Amazon Chime SDK meetings while they're in progress.

Amazon Chime SDK live transcription processes each user's audio separately for improved accuracy in multi-speaker scenarios. The Amazon Chime SDK uses its active talker algorithm to select the top two active talkers, and then sends their audio to Amazon Transcribe, in separate channels, via a single stream. Meeting participants receive user-attributed transcriptions via Amazon Chime SDK data messages. You can use transcriptions in a variety of ways, such as displaying subtitles, creating meeting transcripts, or using the transcriptions for content analysis.

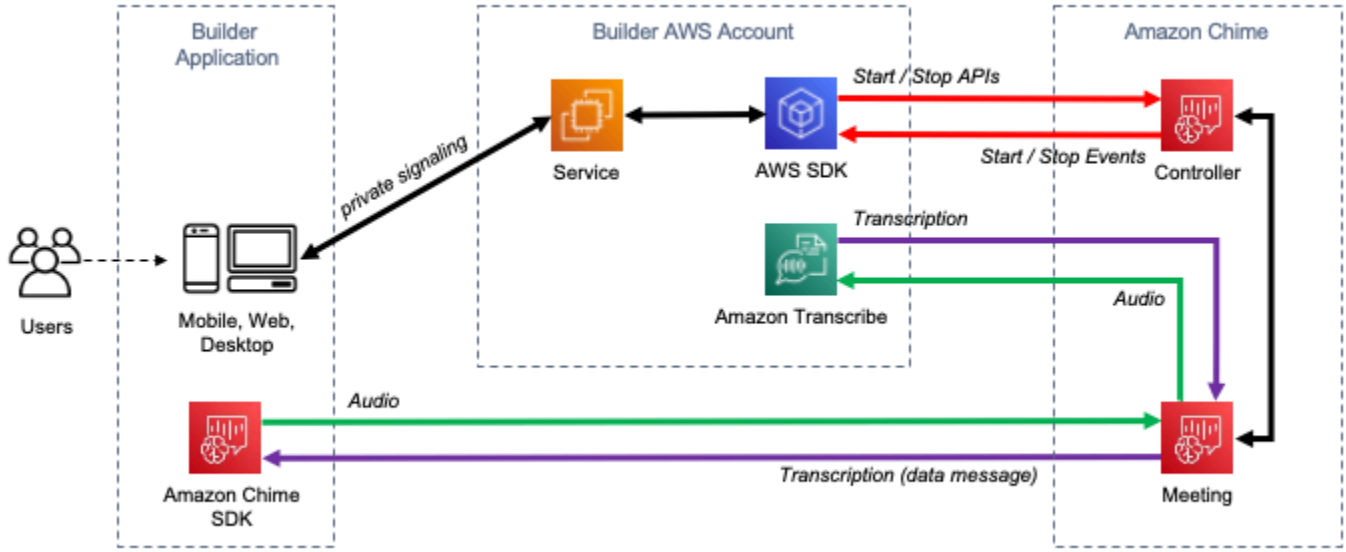
Live transcription uses one stream to Amazon Transcribe for the duration of the meeting transcription. Standard Amazon Transcribe and Amazon Transcribe Medical costs apply. For more information, refer to Amazon Transcribe Pricing. For questions about usage or billing, contact your AWS account manager.

> **Important**
>
> By default, Amazon Transcribe may use and store audio content processed by the service to develop and improve AWS AI/ML services as further described in section 50 of the AWS Service Terms. Using Amazon Transcribe may be subject to federal and state laws or regulations regarding the recording or interception of electronic communications. It is your and your end users' responsibility to comply with all applicable laws regarding the recording, including properly notifying all participants in a recorded session or communication that the session or communication is being recorded, and obtaining all necessary consents. You can opt out from AWS using audio content to develop and improve AWS AI/ML services by configuring an AI services opt out policy using AWS Organizations.

Topics

System architecture

The Amazon Chime SDK creates real-time meeting transcriptions, without audio leaving the AWS network, via a service-side integration with your Amazon Transcribe or Amazon Transcribe Medical account. For improved accuracy, users' audio is processed separately, then mixed into the meeting. The Amazon Chime SDK uses its active talker algorithm to select the top two active talkers, and then sends their audio to Amazon Transcribe or Amazon Transcribe Medical in separate channels via a single stream. For reduced latency, user-attributed transcriptions are sent directly to every meeting participant via data messages. When using a media pipeline to capture meeting audio, the meeting's transcription information is also captured.



Billing and usage

Live transcription uses one stream to Amazon Transcribe or Amazon Transcribe Medical for the duration of the meeting transcription. Standard Amazon Transcribe and Amazon Transcribe Medical costs apply. For more information, see Amazon Transcribe Pricing.. For questions about usage or billing, contact your AWS account manager.

Configuring your account

Before you can use Amazon Chime SDK live transcription, you must grant Amazon Chime SDK permission to call Amazon Transcribe and Amazon Transcribe Medical in your AWS account. You do that by adding the Chime Transcription service-linked role to your account. For information about creating the service-linked role for live transcription, refer to Using roles with live transcription in the *Amazon Chime SDK Administration Guide*. For more information about IAM service-linked roles, refer to Service Linked Roles in the *IAM User Guide*.

Choosing transcription options

When you use Amazon Chime SDK live transcription, you use Amazon Transcribe or Amazon Transcribe Medical in your AWS account. You have access to all the streaming languages supported by Amazon Transcribe, plus features such as custom vocabularies and vocabulary filters. When using Amazon Transcribe Medical, you can choose a medical specialty, conversation type, and

optionally provide any custom vocabulary. Standard Amazon Transcribe and Amazon Transcribe Medical costs apply.

The process of choosing transcription options follows these steps.

Step 1: Choosing a transcription service

You need to decide which transcription service to use, Amazon Transcribe or Amazon Transcribe Medical.

If your use case requires medical speech to text capabilities, you probably want to use Amazon Transcribe Medical. For all other use cases, you probably want to use Amazon Transcribe.

You specify which transcription service to use when you call the `StartMeetingTranscription` API:

- To use Amazon Transcribe, specify a `TranscriptionConfiguration` with `EngineTranscribeSettings`.
- To use Amazon Transcribe Medical, specify a `TranscriptionConfiguration` with `EngineTranscribeMedicalSettings`.

Step 2: Choosing a transcription Region

You need to choose an AWS Region for the transcription service. For information about the available AWS Regions for Amazon Transcribe and Amazon Transcribe Medical, refer to the AWS Regional Services table.

In general, the lowest latency between a meeting's media Region and transcription Region provides the best user experience. For the lowest latency, use the same Region for media and transcription whenever possible. However, you may have other factors to consider in selecting a Region, such as regulatory requirements or the Regions where you have configured Amazon Transcribe or Amazon Transcribe Medical.

Amazon Transcribe and Amazon Transcribe Medical features, such as custom vocabularies or vocabulary filters, are Region specific. If you configure any of those features, you should do so identically in all of the AWS Regions in which you intend to use live transcription. Alternately, you can use the same Amazon Transcribe Region for all meetings.

You can specify the Region that the transcription service uses. You do that by adding the region name to the `Region` field of the transcription engine settings when you call the

`StartMeetingTranscription` API. If you do not specify a Region, the Amazon Chime SDK attempts to use the transcription service in the meeting's media region. To have the Amazon Chime SDK select the Region for the transcription service for you, specify `auto` in the `Region` field. When you do that, Amazon Chime selects the transcription service Region based the meeting's media Region as described in the tables below. For more information about the `StartMeetingTranscription` API, refer to Starting and stopping transcription in this guide.

> **Note**
>
> The transcription region selected by the Amazon Chime SDK is subject to change as AWS, Amazon Chime SDK, Amazon Transcribe and Amazon Transcribe Medical make more regions available.

Automatic region selection for Amazon Transcribe

Amazon Chime SDK Media Region	Region code	Transcription Region
US East (Ohio)	us-east-2	us-east-2
US East (N. Virginia)	us-east-1	us-east-1
US West (N. California)	us-west-1	us-west-2
US West (Oregon)	us-west-2	us-west-2
Africa (Cape Town)*	af-south-1	eu-west-2
Asia Pacific (Mumbai)	ap-south-1	eu-west-2
Asia Pacific (Seoul)	ap-northeast-2	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1	ap-northeast-1
Asia Pacific (Sydney)	ap-southeast-2	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1	ap-northeast-1

Amazon Chime SDK Media Region	Region code	Transcription Region
Canada (Central)	ca-central-1	ca-central-1
Europe (Frankfurt)	eu-central-1	eu-central-1
Europe (Ireland)	eu-west-1	eu-west-1
Europe (London)	eu-west-2	eu-west-2
Europe (Milan)*	eu-south-1	eu-central-1
Europe (Paris)	eu-west-3	eu-central-1
Europe (Stockholm)	eu-north-1	eu-central-1
South America (São Paulo)	sa-east-1	sa-east-1
GovCloud (US-East)	us-gov-east-1	us-gov-west-1
GovCloud (US-West)	us-gov-west-1	us-gov-west-1

Automatic region selection for Amazon Transcribe Medical

Amazon Chime SDK Media Region	Region code	Transcription Region
US East (Ohio)	us-east-2	us-east-2
US East (N. Virginia)	us-east-1	us-east-1
US West (N. California)	us-west-1	us-west-2
US West (Oregon)	us-west-2	us-west-2
Africa (Cape Town)*	af-south-1	eu-west-1
Asia Pacific (Mumbai)	ap-south-1	eu-west-1

Amazon Chime SDK Media Region	Region code	Transcription Region
Asia Pacific (Seoul)	ap-northeast-2	us-west-2
Asia Pacific (Singapore)	ap-southeast-1	ap-southeast-2
Asia Pacific (Sydney)	ap-southeast-2	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1	us-west-2
Canada (Central)	ca-central-1	ca-central-1
Europe (Frankfurt)	eu-central-1	eu-west-1
Europe (Ireland)	eu-west-1	eu-west-1
Europe (London)	eu-west-2	us-east-1
Europe (Milan)*	eu-south-1	eu-west-1
Europe (Paris)	eu-west-3	eu-west-1
Europe (Stockholm)	eu-north-1	eu-west-1
South America (São Paulo)	sa-east-1	us-east-1

> **Note**
>
> To use live transcription in Regions marked with an asterisk (*), you must first enable the Region in your AWS account. For more information, refer to Enabling a Region in the AWS General Reference.

For more information about the regions and endpoints for each service, refer to:

- Amazon Chime SDK media Regions
- Amazon Transcribe endpoints and quotas
- Amazon Transcribe Medical endpoints and quotas

Step 3: Review service quotas

Each Amazon Chime SDK meeting with live transcription requires exactly one HTTP/2 stream to Amazon Transcribe or Amazon Transcribe Medical. Both services have regional service quotas for the number of concurrent HTTP/2 streams, and for start-stream transactions per second. For more information about the quotas, refer to Guidelines and quotas in the *Amazon Transcribe Developer Guide*. For information about quota increases, refer to Service Quotas in the AWS console.

Starting and stopping transcription

You use the Amazon Chime SDK StartMeetingTranscription API to initiate meeting transcription by applying a `TranscriptionConfiguration` to the meeting. The Amazon Chime SDK controller forwards the configuration to the meeting asynchronously. The success or failure of initiating meeting transcription is signaled through a message via Amazon Simple Notification Service (Amazon SNS) and Amazon EventBridge.

Starting transcription

This example shows how to start live transcription with Amazon Transcribe.

```
POST /meetings/meetingId/transcription?operation=start HTTP/1.1
Content-type: application/json
{
    "TranscriptionConfiguration": {
        "EngineTranscribeSettings": {
            "LanguageCode": "en-US",
            "VocabularyFilterMethod": "tag",
            "VocabularyFilterName": "profanity",
            "VocabularyName": "lingo",
            "Region": "us-east-1"
            "EnablePartialResultsStabilization": true,
            "PartialResultsStability": "high",
            "ContentIdentificationType": "PII",
            "ContentRedactionType": "PII",
            "PiiEntityTypes": "ALL",
            "LanguageModelName": "language-model"
        }
    }
}
```

This example shows how to start live transcription with Amazon Transcribe Medical.

```
POST /meetings/meetingId/transcription?operation=start HTTP/1.1
Content-type: application/json
{
    "TranscriptionConfiguration": {
        "EngineTranscribeMedicalSettings": {
            "LanguageCode": "en-US",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION",
            "VocabularyName": "lingo",
            "Region": "us-east-1",
           "ContentIdentificationType": "PHI",
        }
   }
}
```

`StartMeetingTranscription` – Starts transcription for the meeting.

- `meetingId` – The ID of the meeting, returned by the CreateMeeting API.
- `TranscriptionConfiguration` – Encapsulates the parameters for live transcription. You must specify exactly one configuration, `EngineTranscribeSettings` or `EngineTranscribeMedicalSettings`.

`EngineTranscribeSettings` – Specifies the use of Amazon Transcribe and passes its settings through to Amazon Transcribe.

- `LanguageCode` – Required.
- `VocabularyFilterMethod` – Optional.
- `VocabularyFilterName` – Optional.
- `VocabularyName` – Optional.
- `Region` – Optional.
- `EnablePartialResultsStabilization` – Optional.
- `PartialResultsStability` – Optional.
- `ContentIdentificationType` – Optional.
- `ContentRedactionType` – Optional.

`PiiEntityTypes` – Optional.

`LanguageModelName` – Optional.

`EngineTranscribeMedicalSettings` – Specifies the use of Amazon Transcribe Medical and passes its settings through to Amazon Transcribe Medical.

`LanguageCode` – Required.

`Speciality` – Required.

`Type` – Required.

`VocabularyName` – Optional.

`Region` – Optional.

`ContentIdentificationType` – Optional.

Responses

Amazon Transcribe and Amazon Transcribe Medical take the following responses:

- OK (200) with empty body, if you successfully apply the `TranscriptionConfiguration` to the meeting.

Error messages

Amazon Transcribe and Amazon Transcribe Medical display the following error messages:

- **BadRequestException (400):** The input parameters don't match the service's restrictions.
- **ForbiddenException (403):** The client is permanently forbidden from making the request.
- **NotFoundException (404):** The `meetingId` does not exist.
- **ResourceLimitExceededException (400):** The request exceeds the resource limit. For example, too many meetings have live transcription enabled.
- **ServiceFailureException (500):** The service encountered an unexpected error.
- **ServiceUnavailableException (503):** The service is currently unavailable.
- **ThrottledClientException (429):** The client exceeded its request rate limit.
- **UnauthorizedClientException (401):** The client is not currently authorized to make the request.

Calling `StartMeetingTranscription` a second time updates the `TranscriptionConfiguration` applied to the meeting.

Stopping transcription

You use the StopMeetingTranscription API to remove the `TranscriptionConfiguration` for a given `meetingID` and end meeting transcription. Ending a meeting stops transcription automatically.

This example shows the request syntax that invokes `StopMeetingTranscription`.

```
POST/meetings/meetingId/transcription?operation=stop HTTP/1.1
```

Responses

Amazon Transcribe and Amazon Transcribe Medical take the following responses:

- OK (200) with empty body, if you successfully remove the `TranscriptionConfiguration` from the meeting.

Error messages

Amazon Transcribe and Amazon Transcribe Medical display the following error messages:

- **BadRequestException (400):** The input parameters don't match the service's restrictions.
- **ForbiddenException (403):** The client is permanently forbidden from making the request.
- **NotFoundException (404):** The `meetingId` does not exist.
- **ServiceFailureException (500):** The service encountered an unexpected error.
- **ServiceUnavailableException (503):** The service is currently unavailable.
- **ThrottledClientException (429):** The client exceeded its request rate limit.
- **UnauthorizedClientException (401):** The client is not currently authorized to make the request.

Transcription parameters

The Amazon Transcribe and Amazon Transcribe Medical APIs offer a number of parameters when initiating streaming transcription, such as StartStreamTranscription and StartMedicalStreamTranscription. You can use t hose parameters in the

`StartMeetingTranscription` API unless the Amazon Chime SDK predetermines the parameter's value. For example, the `MediaEncoding` and `MediaSampleRateHertz` parameters are not available because the Amazon Chime SDK sets them automatically.

Amazon Transcribe and Amazon Transcribe Medical validate the parameters, and that allows you to use new parameter values as soon as they become available. For example, if Amazon Transcribe Medical launches support for a new language, you only need to specify the new language value in the `LanguageCode` parameter.

Understanding transcription events

The Amazon Chime SDK sends transcription lifecycle events, which you can use to trigger notifications and initiate downstream work flows. Some examples of using transcription events include:

- Measuring the adoption of live transcription in Amazon Chime SDK meetings
- Tracking language preferences

You can send events to Amazon EventBridge, Amazon Simple Notification Service, and Amazon Simple Queue Service. For more information, refer to Events from AWS services in the *Amazon EventBridge User Guide*.

Amazon Chime SDK meeting transcription started

The Amazon Chime SDK sends this event when meeting transcription is started or the TranscriptionConfiguration is updated.

Example: Event data

The following is example data for this event.

```
{
    "version": "0",
    "source": "aws.chime",
    "account": "111122223333",
    "id": "12345678-1234-1234-1234-111122223333",
    "region": "us-east-1",
    "detail-type": "Chime Meeting State Change",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": []
    "detail": {
```

```
        "version": "0",
        "eventType": "chime:TranscriptionStarted",
        "timestamp": 12344566754,
        "meetingId": "87654321-4321-4321-1234-111122223333",
        "externalMeetingId": "mymeeting",
        "mediaRegion": "us-west-1",
        "transcriptionRegion": "us-west-2",
        "transcriptionConfiguration": "{...}"
    }
}
```

Amazon Chime SDK meeting transcription stopped

The Amazon Chime SDK sends this event when meeting transcription is stopped.

Example: Event data

The following is example data for this event.

```
{
    "version": "0",
    "source": "aws.chime",
    "account": "111122223333",
    "id": "12345678-1234-1234-1234-111122223333",
    "region": "us-east-1",
    "detail-type": "Chime Meeting State Change",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": []
    "detail": {
        "version": "0",
        "eventType": "chime:TranscriptionStopped",
        "timestamp": 12344566754,
        "meetingId": "87654321-4321-4321-1234-111122223333",
        "externalMeetingId": "mymeeting",
        "mediaRegion": "us-west-1",
        "transcriptionRegion": "us-west-2",
        "transcriptionConfiguration": "{...}"
    }
}
```

Amazon Chime SDK meeting transcription interrupted

The Amazon Chime SDK sends this event if meeting transcription is interrupted.

Example: Event data

The following is example data for this event.

```
{
    "version": "0",
    "source": "aws.chime",
    "account": "111122223333",
    "id": "12345678-1234-1234-1234-111122223333",
    "region": "us-east-1",
    "detail-type": "Chime Meeting State Change",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": []
    "detail": {
        "version": "0",
        "eventType": "chime:TranscriptionInterrupted",
        "timestamp": 12344566754,
        "meetingId": "87654321-4321-4321-1234-111122223333",
        "externalMeetingId": "mymeeting",
        "message": "Internal server error",
        "mediaRegion": "us-west-1",
        "transcriptionRegion": "us-west-2",
        "transcriptionConfiguration": "{...}"
    }
}
```

Amazon Chime SDK meeting transcription resumed

The Amazon Chime SDK sends this event if meeting transcription is resumed after an interruption.

Example: Event data

The following is example data for this event.

```
{
    "version": "0",
    "source": "aws.chime",
    "account": "111122223333",
    "id": "12345678-1234-1234-1234-111122223333",
    "region": "us-east-1",
    "detail-type": "Chime Meeting State Change",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": []
```

```
    "detail": {
        "version": "0",
        "eventType": "chime:TranscriptionResumed",
        "timestamp": 12344566754,
        "meetingId": "87654321-4321-4321-1234-111122223333",
        "externalMeetingId": "mymeeting",
        "mediaRegion": "us-west-1",
        "transcriptionRegion": "us-west-2",
        "transcriptionConfiguration": "{...}"
    }
}
```

Amazon Chime SDK meeting transcription failed

The Amazon Chime SDK sends this event if meeting transcription failed to start, or failed to resume after an interruption.

Example: Event data

The following is example data for this event.

```
{
    "version": "0",
    "source": "aws.chime",
    "account": "111122223333",
    "id": "12345678-1234-1234-1234-111122223333",
    "region": "us-east-1",
    "detail-type": "Chime Meeting State Change",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": []
    "detail": {
        "version": "0",
        "eventType": "chime:TranscriptionFailed",
        "timestamp": 12344566754,
        "meetingId": "87654321-4321-4321-1234-111122223333",
        "externalMeetingId": "mymeeting",
        "message": "Internal server error",
        "mediaRegion": "us-west-1",
        "transcriptionRegion": "us-west-2",
        "transcriptionConfiguration": "{...}"
    }
}
```

Understanding transcription messages

The Amazon Chime SDK service shares transcription information with attendees by sending `TranscriptEvent` objects in data messages. A `TranscriptEvent` delivers a `Transcript` or a `TranscriptionStatus`.

A `Transcript` includes results with timestamped, user-attributed words and punctuation. A result may be “partial”, in which case the system usually updates it in a subsequent `TranscriptEvent`. This allows you to see transcriptions quickly and apply inline updates later as necessary.

A `TranscriptStatus` may deliver one of the `TranscriptionStatusType` events, listed in the example in the next section.

Newer versions of the Amazon Chime SDKs include additional data types and helper functions for common processing a `TranscriptEvent`.

TranscriptEvent

This example shows a typical transcription event.

```
type TranscriptEvent = Transcript | TranscriptionStatus;

export class TranscriptEventConverter {
  static from(dataMessage: DataMessage): TranscriptEvent[] {
    // convert DataMessage to TranscriptEvents
    return ...
  }
}

export default class TranscriptionStatus {
    type: TranscriptionStatusType;
    eventTimeMs:                   number;
    transcriptionRegion:           string;
    transcriptionConfiguration:    string;
    message?:                      string;
}

enum TranscriptionStatusType {
    STARTED        =    'started',
    INTERRUPTED    =    'interrupted',
    RESUMED        =    'resumed',
    STOPPED        =    'stopped',
    FAILED         =    'failed',
```

```
}

export default class Transcript {
    results: TranscriptResult[];    // at least one
}

export class TranscriptResult {
    resultId:        string;
    isPartial:       boolean;
    startTimeMs:     number;
    endTimeMs:       number;
    alternatives:    TranscriptAlternative[];    // most confident first
    }

export default class TranscriptAlternative {
    items: TranscriptItem[];    // in start time order
    transcript: string; //concatenated transcript items
    entities?: TranscriptEntity[];
}

export default class TranscriptItem {
    type:                      TranscriptItemType;
    startTimeMs:               number;
    endTimeMs:                 number;
    attendee:                  Attendee;
    content:                   string;
    vocabularyFilterMatch?:    boolean;
    confidence?:               number;
    stable?:                   boolean;
}

enum TranscriptItemType {
    PRONUNCIATION    =    'pronunciation',// content is a word
    PUNCTUATION      =    'punctuation',// content is punctuation
}

export default class TranscriptEntity {
    category:       string;
    confidence:     number;
    content:        string;
    endTimeMs:      number;
    startTimeMs:    number;
    type?:          string;
}
```

```
// This is an existing SDK model
export default class Attendee {
    attendeeId:        string;
    externalUserId:    string;
}
```

Data guidelines

Keep these guidelines in mind as you go.

1. `transcription.results` may have more than one result.
2. If `transcription.results[i].isPartial = true`, then there may be an update for the entire result. The update is likely, but not guaranteed. The update has the same `transcript.result[i].resultId`. If you want to avoid low-confidence transcriptions, you can skip partial results completely. If you want low-latency results, you can display partial results, then overwrite completely when the update arrives.
3. `transcription.results[i].alternatives` always contains at least one entry. If it contains more than one entry, the most confident entry is first in the list. In most cases, you can take the first entry in `transcription.results[i].alternatives` and ignore the others.
4. `transcription.results[i].alternatives[j].items` includes an entry for each word or punctuation mark.
5. `transcription.results[i].alternatives[j].items[k].` content is what was spoken.
6. `transcription.results[i].alternatives[j].items[k].attendee` is the user attribution (who) of the content.
7. `transcription.results[i].alternatives[j].items[k].startTimeMs` is the "when" of the content. This enables word-by-word rendering of user-attributed transcription across different users in the order that the words were spoken.
8. The `transcription.results[i].alternatives[j].items[k].endTimeMs` field can generally be ignored, but is supplied for completeness of who said what when.
9. `transcription.results[i].alternatives[j].items[k].vocabularyFilterMatch` is true if the content matched a word in the filter, otherwise it is false.
10. `transcription.results[i].alternatives[j].items[k].confidence` is a value between 0 and 1. It indicates the engine's confidence that the item content correctly matches the spoken word, with 0 being the lowest confidence and 1 being the highest confidence.

11. `transcription.results[i].alternatives[j].items[k].stable` indicates whether the current word will change in future partial result updates. This value can only be true if you enable the partial results stabilization feature by setting `EnablePartialResultsStabilization` to `true` in your request.
12. `transcription.results[i].alternatives[j].entities` includes an entry for each entity that the Content Identification or Redaction features detect. The list is only populated if you enable Content Identification or Redaction. An entity can be data such as personally identifiable information or personal health information. You can use entities to highlight, or take action on, words of interest during transcription.
13. `transcription.results[i].alternatives[j].entities[k].category` is the entity's category. It equals the Content Identification or Redaction type, such as "PII" or "PHI", which is provided in the request.
14. `transcription.results[i].alternatives[j].entities[k].confidence` measures how strong the engine is that the particular content is truly an entity. Note that this is different than the item-level confidence, which measures how confident the engine is in the correctness of the words themselves.
15. `transcription.results[i].alternatives[j].entities[k].content` is the actual text that makes up the entity. This can be multiple items, such as an address.
16. `transcription.results[i].alternatives[j].entities[k].startTimeMs` captures the time at which the entity started being spoken.
17. `transcription.results[i].alternatives[j].entities[k].endTimeMs` captures the time at which the entity finished being spoken.
18. `transcription.results[i].alternatives[j].entities[k].type` is only supported for the Transcribe engine and provides the sub-type of the entity. These are values such as `ADDRESS`, `CREDIT_DEBIT_NUMBER`, and so on.

Registering event handlers for TranscriptEvents

The following examples use the Amazon Chime SDK client library for JavaScript. However, the pattern is consistent across all Amazon Chime SDKs.

The `TranscriptionController` in the `RealtimeController` and `RealtimeControllerFacade` includes specific functions for adding a handler that processes `TranscriptionEvents`:

```
/**
```

```
 * Returns the [[TranscriptionController]] for this real-time controller.
 */
readonly transcriptionController?: TranscriptionController;
```

The `TranscriptionController` has two functions to manage subscribing and unsubscribing to `TranscriptionEvent` callbacks:

```
import TranscriptEvent from './TranscriptEvent';

export default interface TranscriptionController {
  /**
   * Subscribe a callback to handle received transcript event
   */
  subscribeToTranscriptEvent(callback: (transcriptEvent: TranscriptEvent) => void):
 void;

  /**
   * Unsubscribe a callback from receiving transcript event
   */
  unsubscribeFromTranscriptEvent(callback: (transcriptEvent: TranscriptEvent) => void):
 void;
}
```

Using the optional `TranscriptionController`

We provide a default implementation of `TranscriptionController` interface named `DefaultTranscriptionController`. The default implementation in `DefaultRealtimeController` and `DefaultAudioVideoFacade` returns a `DefaultTranscriptionController` object:

```
/**
get transcriptionController(): TranscriptionController {
   return this.realtimeController.transcriptionController;
}
```

`DefaultRealtimeController` also takes an optional `TranscriptionController` object in its constructor. That allows you to override the `DefaultTranscriptionController` behavior. Developer applications subscribe and unsubscribe to one or more callbacks through the `TranscriptionController` object of the `AudioVideoFacade` object:

```
// Subscribe
```

```
this.audioVideo.transcriptionController?.subscribeToTranscriptEvent(this.transcriptEventHandler

// Unsubscribe
this.audioVideo.transcriptionController?.unsubscribeFromTranscriptEvent(this.transcriptEventHan
```

Processing a received transcript event

The following examples show how to process a received `TranscriptEvent`.

> **Note**
>
> The exact output depends on several factors, including how quickly individuals talk and when they pause.

Example 1: StartMeetingTranscription

This example shows a typical `StartMeetingTranscription` operation.

```
meeting.StartMeetingTranscription(
    { EngineTranscribeSettings: { Languagecode: ‘en-US’ } } );
```

The operation generates a `TranscriptEvent`.

```
{
    status: {
        type: 'started',
        eventTimeMs: 1620118800000,
        transcriptionConfig: {
            LanguageCode: 'en-US'
        }
    }
}
```

Example 2: A partial transcript result

In this example, an attendee says, "The quick brown fox jumps over the lazy dog." Note that in this example, the `isPartial` value is `true`. If you look deeper into the message, you can see that the system processed the word "fox" as "facts." The system uses the same `resultId` to update the transcript.

```
{
    transcript: {
        results: [{
            resultId:"1",                                isPartial: true,
            startTimeMs: 1620118800000,                  endTimeMs: 1620118801000,
            alternatives: [{
                items:[{
                    type:        'pronunciation',
                    startTimeMs: 1620118800000,          endTimeMs: 1620118800200,
                    attendee: { attendeeId: "1",         externalUserId: "A"},
                    content: "the",                      vocabularyFilterMatch: false
                },
                {
                    type:        'pronunciation',
                    startTimeMs: 1620118800200,           endTimeMs: 1620118800400,
                    attendee: { attendeeId: "1",          externalUserId: "A" },
                    content:"quick",                      vocabularyFilterMatch: false
                },
                {
                    type:'pronunciation',
                    startTimeMs: 1620118800400,           endTimeMs: 1620118800750,
                    attendee: { attendeeId: "1",          externalUserId: "A" },
                    content:"brown",                      vocabularyFilterMatch: false
                },
                {
                    type:'pronunciation',
                    startTimeMs: 1620118800750,           endTimeMs: 1620118801000,
                    attendee:{ attendeeId: "1",           externalUserId: "A" },
                    content:"facts",                      vocabularyFilterMatch: false
                },
                {
                    type:'punctuation',
                    startTimeMs: 1620118801000,           endTimeMs: 1620118801500,
                    attendee:{ attendeeId: "1",           externalUserId: "A" },
                    content:    ",",                      vocabularyFilterMatch: false
                }]
            }]
        }]
    }
}
```

Example 3: A final transcript result

In the event of a partial transcript, the system processes the phrase again. This example has an `isPartial` value of `false`, and the message contains "fox" instead of "facts." The system re-issues the message using the same ID.

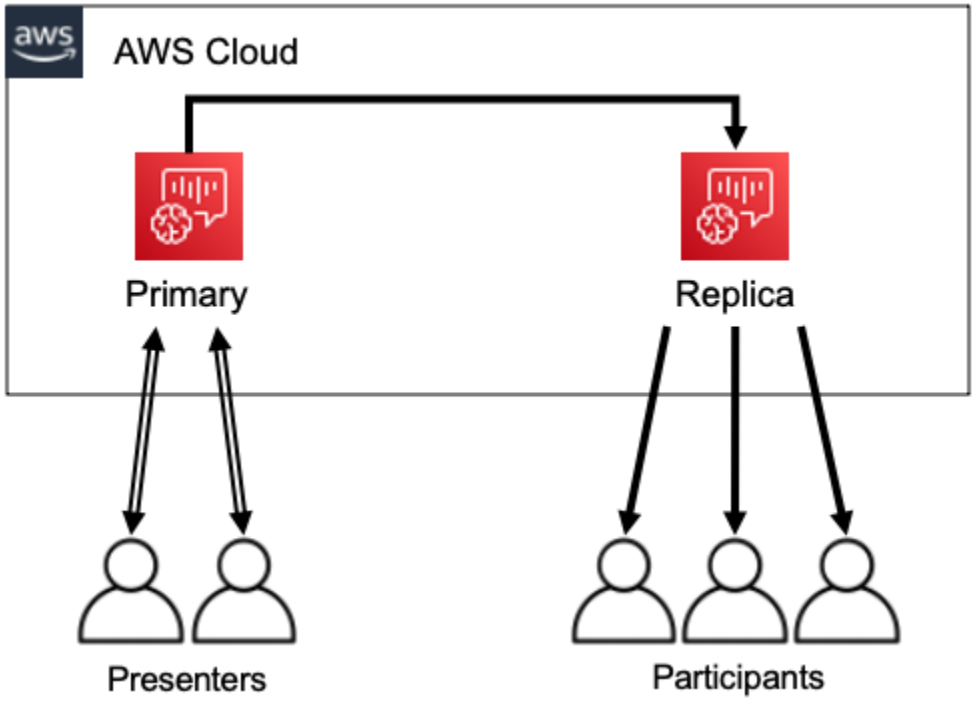
```
{
    transcript: {
        results: [{
            resultId:"1",                                isPartial: false,
            startTimeMs: 1620118800000,                  endTimeMs: 1620118801000,
            alternatives: [{
                items:[{
                    type:        'pronunciation',
                    startTimeMs: 1620118800000,          endTimeMs: 1620118800200,
                    attendee: { attendeeId: "1",         externalUserId: "A"},
                    content: "the",                      vocabularyFilterMatch: false
                },
                {
                    type:        'pronunciation',
                    startTimeMs: 1620118800200,          endTimeMs: 1620118800400,
                    attendee: { attendeeId: "1",         externalUserId: "A" },
                    content:"quick",                     vocabularyFilterMatch: false
                },
                {
                    type:'pronunciation',
                    startTimeMs: 1620118800400,          endTimeMs: 1620118800750,
                    attendee: { attendeeId: "1",         externalUserId: "A" },
                    content:"brown",                     vocabularyFilterMatch: false
                },
                {
                    type:'pronunciation',
                    startTimeMs: 1620118800750,          endTimeMs: 1620118801000,
                    attendee: { attendeeId: "1",          externalUserId: "A" },
                    content:"fox",                       vocabularyFilterMatch: false
                },
                {
                    type:'punctuation',
                    startTimeMs: 1620118801000,          endTimeMs: 1620118801500,
                    attendee: { attendeeId: "1",          externalUserId: "A" },
                    content:    ",",                     vocabularyFilterMatch: false
                }]
            }]
```

```
        }]
    }
}
```

Using media replication

You can use media replication to link a primary WebRTC session with multiple replica sessions to reach larger audiences. Each WebRTC media session supports 250 connections, and you can replicate a primary session to multiple replica sessions. Participants connected to a replica session receive only the audio and video of the presenters connected to the primary session. They have no knowledge of the participants connected to the replicated session, which makes media replication ideal for webinars and other use cases where privacy is desired.

The following image shows media replication between a primary session with presenters sharing audio and video, and a replica session with participants consuming the media.



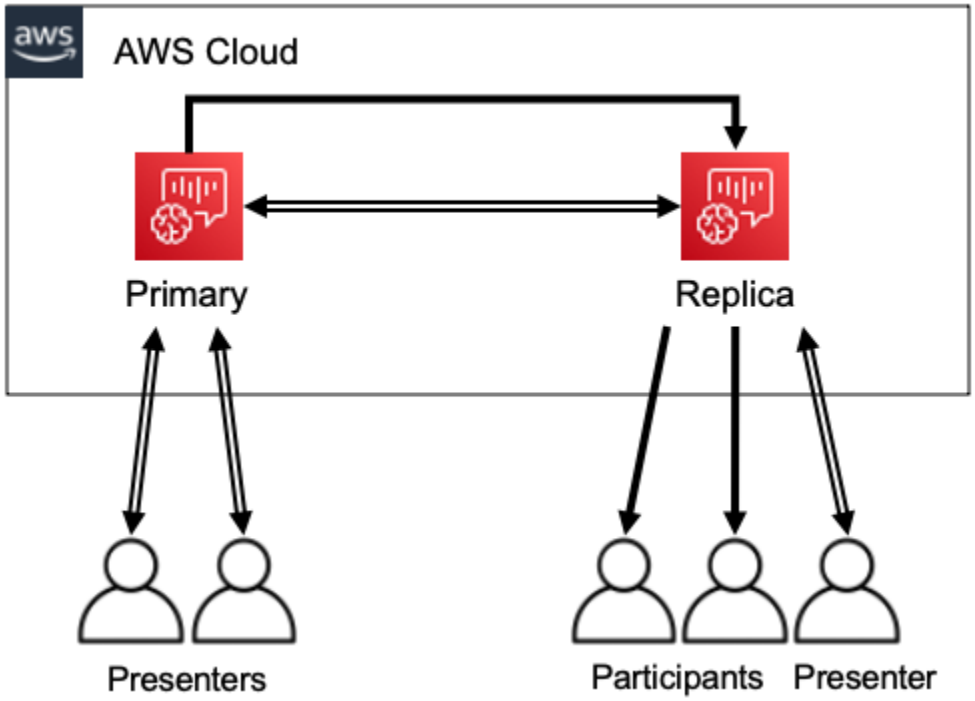
> **Note**
>
> The service quota *Chime SDK Meetings - replica meetings per primary meeting* has a default value of 4, and you can increase that limit on request. For more information about quotas, refer to AWS service quotas in the *AWS General Reference*.

Topics

Interactive participants

Participants connected to a replica session can be granted access to join the primary session. Because everyone uses a WebRTC connection, presenters and participants don't experience transcoding delays. When participants switch between primary and replicated sessions, they reuse their WebRTC connections, so switching is extremely fast. That allows participants to contribute to the live conversation without missing any content.

The following image shows a participant in a replica session using their WebRTC connection to switch to the primary session.

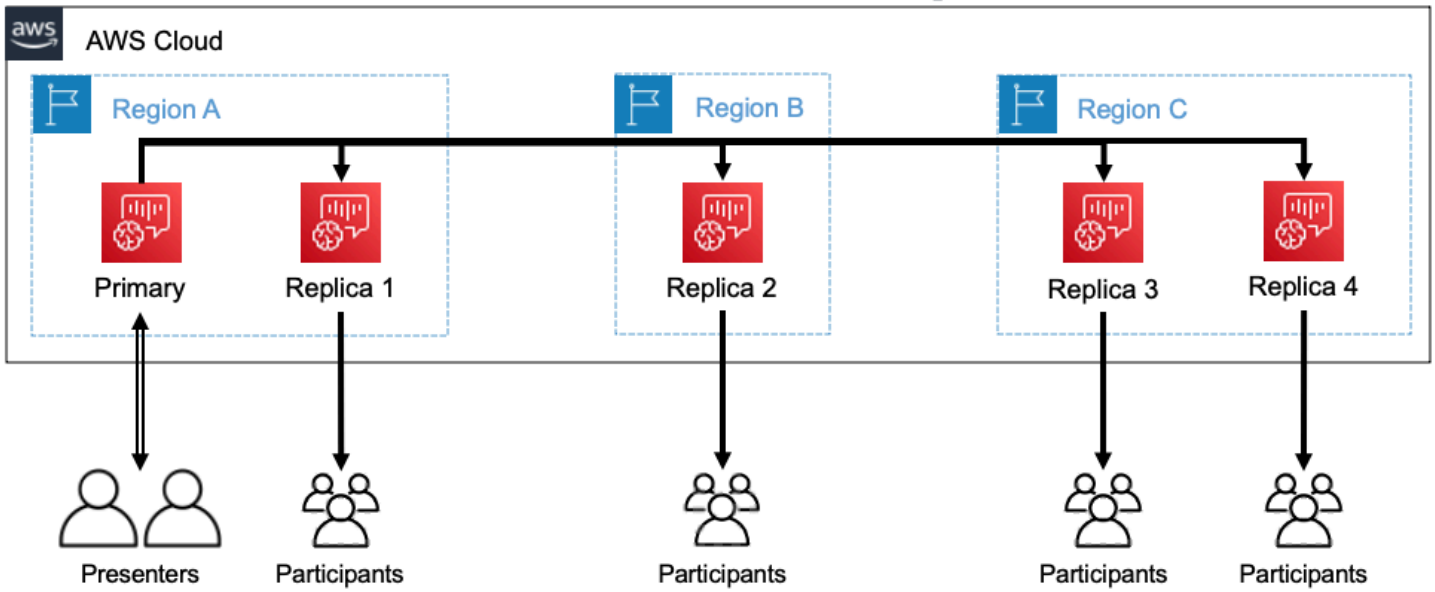


Global participants

You can choose the AWS Region for each WebRTC media session. That allows you to create replica sessions in Regions closer to your participants than the primary session's Region. When you do this, media flows from the primary session to the replica sessions across the AWS network, then from

the replica session to the participant across the Internet. When presenting to a global audience, having replica sessions near your participants can help ensure that media travels around the world on the AWS network, instead of the internet, for a better meeting experience.

The following image shows a primary session and replicated sessions in different Regions.



Session lifecycle

Creating sessions

You use the CreateMeeting or CreateMeetingWithAttendees APIs to create WebRTC media sessions. By default, the APIs create a primary session unless you specifically create a replica session.

You create a replica session by specifying the `MeetingId` of the primary session as the `PrimaryMeetingId` in the `CreateMeeting` or `CreateMeetingWithAttendees` API call.

> **Note**
>
> If you specify the `MeetingId` of a replica session as the `PrimaryMeetingId`, the API call will fail.

Creating attendees

To create the attendee credentials needed to join a WebRTC media session, you can use the CreateMeetingWithAttendees, BatchCreateAttendee, or CreateAttendee APIs.

> **Note**
>
> When creating sessions for a large number of attendees, use `CreateMeetingWithAttendees` or `BatchCreateAttendee` to minimize the number of API calls required.

Deleting attendees

You use the DeleteAttendee API to revoke a attendee's credentials for a WebRTC media session. If the attendee is connected to the session, they will disconnected and cannot rejoin.

When you use the DeleteMeeting API to delete a WebRTC media session, the API automatically deletes all attendees and you don't need to call `DeleteAttendee`.

Switching sessions

To allow a participant to switch from a replica session to a primary session, you must create credentials for them in the primary meeting. Refer to *Creating attendees* earlier in this list. Use the credentials with the `promoteToPrimaryMeeting` method in the Amazon Chime SDK client library to switch to the primary session.

To switch participants back to a replica session, use the `demoteFromPrimaryMeeting` method in the Amazon Chime SDK client library, or use the DeleteAttendee API to invalidate their primary session credentials.

> **Note**
>
> A presenter who connects directly to a primary session can't switch to a replica session.

For more information on switching between sessions, refer to the client library documentation:

- Amazon Chime SDK for Android on GitHub.
- Amazon Chime SDK for iOS on GitHub.
- Amazon Chime SDK client library for JavaScript on GitHub.

Deleting sessions

You use the DeleteMeeting API to delete WebRTC media sessions.

If you delete a primary session, the `DeleteMeeting` API automatically deletes all attached replica sessions. So to delete all sessions, just delete the primary.

The service automatically deletes a primary session if no attendees connect for 5 contiguous minutes. The service only deletes replica sessions automatically when it deletes a primary session. That means you can create replica sessions when you create a primary session, and the replicas will be available for the duration of the primary session.

Troubleshooting and debugging Amazon Chime SDK meetings

Use the following topics to help diagnose and troubleshoot issues that you encounter when working with the Amazon Chime SDK.

Topics

Understanding the system requirements

As part of troubleshooting, make sure you code for supported browsers. For a current list of supported browsers, versions, and operating systems, see Amazon Chime SDK system requirements. The developer guide and FAQs on Github address browser and other compatibility issues. Also, familiarize yourself with the known browser issues on GitHub and any workarounds.

If you just started with Amazon Chime SDK Meetings, the Amazon Chime SDK's Builder Journey provides a step-by-step guide for building with the Amazon Chime SDK, plus the tools needed for troubleshooting.

Setting up logging and monitoring

Logging helps you collect information such as server-side meeting events and client-side browser console logs.

The Amazon Chime SDK provides server-side meeting events that you can send to Amazon EventBridge and Amazon CloudWatch Events logs. You can create CloudWatch metrics and insights, and use them in your dashboard for monitoring. The Server-side Logging and Monitoring of Amazon Chime SDK events blog post explains how to enable the CloudWatch Metrics, Insights and Dashboard.

The Amazon Chime SDK provides client-side events for audio and video quality, network bandwidth, and connectivity issues. The Monitoring and troubleshooting with Amazon Chime SDK Meeting events blog post explains how to enable CloudWatch Metrics, Insights and Dashboard for join failures, audio quality issues, and microphone and camera setup failures. For additional information about meeting events, see Meeting Events on Github.

Options for troubleshooting metrics

You have the following options for collecting troubleshooting events.

- Send metrics at every event
- Batch events every N seconds
- Send metrics at end of the meeting
- Logging level for browser console logs

Recommended metrics

At a minimum, you should collect and log the following metrics.

- SDK platform and version
- Browser and version
- Operating system
- Logical cores
- Meeting started
- Meeting ended
- Attendee joined
- Attendee left
- Attendees dropped

Additionally, depending on the issues you face, the following metrics can provide information about connectivity, bandwidth, and quality issues. You can log every occurrence of these metrics, or just count them. Counting can provide a summarized view of the underlying issues:

- connectionDidSuggestStopVideo
- connectionDidBecomeGood
- connectionDidBecomePoor
- Attendee join time > t seconds
- MeetingStartFailed
- MeetingFailed

Enabling client-side logging

You can enable `INFO`-level browser logs by passing `LogLevel.INFO` to the `ConsoleLogger` object.

```
const logger = new ConsoleLogger('MyLogger', LogLevel.INFO);const meetingSession = new
 DefaultMeetingSession(configuration,logger,deviceController);
```

You can also use the `POSTLogger` component in the Amazon Chime SDK for JavaScript to capture browser logs in your back end, such as Amazon CloudWatch Logs. `POSTLogger` makes `HTTP POST` requests to upload browser logs to the given URL in the POSTLogger constructor. For example, the Amazon Chime SDK serverless demo on GitHub uses the `POSTLogger` to send browser logs to Amazon CloudWatch Logs for future investigation.

Enabling server-side logging

The Amazon Chime SDK for JavaScript also calls the `eventDidReceive` observer method with key meeting events, such as `MeetingStartFailed` and `MeetingFailed`. Meeting events often includes specific reasons for failures. For example, say that a large group of customers experience failures. Your web application can collect those meeting events, and then share them with us to troubleshoot the root cause. For more information about meeting events, see the meeting event guide on GitHub, and the Monitoring and troubleshooting with Amazon Chime SDK meeting events blog post.

Troubleshooting Amazon Chime SDK meetings

The sections in this topic explain several ways to self-troubleshoot Amazon Chime SDK meetings.

Topics

Checking FAQs and known issues

Check these FAQs and lists of known issues on GitHub for troubleshooting and debugging advice.

- Amazon Chime SDK – JavaScript - Meetings
- Amazon Chime SDK – JavaScript - Media
- Amazon Chime SDK – JavaScript - Networking
- Amazon Chime SDK – - Audio and Video

Verifying network access

Enterprises often have network firewalls that restrict access to specific ports, or connections to IP address ranges out of their network. The following sections explain some of the ways you can verify network access.

Topics

Validating AWS SDK and Amazon Chime SDK subnets and ports

Applications that use the Amazon Chime SDK utilize two tiers, server and client. The server tier uses the AWS SDK and has server-side meeting handlers. The client tier uses client SDKs.

The AWS SDK is used to call server APIs such as CreateMeeting. Such APIs connect to the AWS global service endpoints in the `us-east-1`, `us-west-2`, `ap-southeast-1`, `eu-central-1`, `us-gov-east-1`, and `us-gov-west-1` Regions. The AWS IP address ranges page in the *AWS General Reference* lists the IP address ranges for each Region. For information about service endpoints and quotas, see the Amazon Chime SDK endpoints and quotas.

Client SDKs, such as the Amazon Chime SDK for JavaScript, connect to service endpoints in the `*.chime.aws` domain.

Use the following validations to ensure that you have network permissions:

- Run the Amazon Chime SDK Meeting Readiness Checker on GitHub to verify that you can reach your network and ports.
- Verify you can resolve *.chime.aws domain from your network or your end user's network.
- Ensure that your firewall allows connections to the AWS IP range via TCP Port 443 for control commands and UDP Port 3478 for media.

Using demo apps to reproduce issues

As a best practice, start the debugging process by trying to reproduce your issue in one of demo apps. That enables the service team to locate where the issue might be. If you can't reproduce the issue with a demo app, you can review the app's code to see how it implemented the relevant use case.

Amazon Chime SDK	Feature	Demo app resources
JavaScript SDK	Meetings	Demo instructions, Source code
React components	Meetings	Demo instructions Source code
Meeting chat	Messaging	Blog post, Demo instructions, Source code
iOS/Android	Meetings	(Blog post) Building a Meeting Application on Android using the Amazon Chime SDK (Blog post) Building a Meeting Application on iOS using the Amazon Chime SDK

Amazon Chime SDK	Feature	Demo app resources
PSTN audio	Inbound calling	Blog post Source code

Using the Meeting Readiness Checker

Use the Amazon Chime SDK Meeting Readiness Checker on GitHub. The checker helps verify audio and video devices, and user connections. You can present the results to your end users with by using pass/fail statues that expose the root cause of any issues.

Understanding common issues with Amazon Chime SDK meetings

The following sections provide troubleshooting methods for common meeting issues.

Topics

- Connectivity issues
- Audio and video quality issues
- Verifying SDK quotas and API throttling
- Opening support cases for Amazon Chime SDK meetings

Connectivity issues

For connectivity issues, see Verifying network access.

Audio and video quality issues

Audio and video quality issues can have several causes. Two main reasons for sub optimal audio/video quality are network bandwidth, and device performance. For detailed information on different challenges and how these impact audio/video quality, see Quality, Bandwidth, and Connectivity on *GitHub*. This article describes different events and metrics that can be monitored to detect bandwidth issues and potential mitigations.

You can choose a Media Region that is closer to the audience of the target meeting session. To understand how to choose an optimal media region, see Using meeting Regions (https://docs.aws.amazon.com/chime-sdk/latest/dg/chime-sdk-meetings-regions.html).

Depending on the bandwidth available for a meeting attendee, the Amazon Chime SDK adjusts the video quality of the video being received/uploaded. To understand how you can control the

video quality for different video layouts, visit Managing Video Quality for different Video Layouts (https://aws.github.io/amazon-chime-sdk-js/modules/videolayout.html). This article describes video lifecycle management and uplink/downlink policies.

Video resolution considerations

- Default resolution for uploading video is 540p and 15fps at 1400 kbps. Depending on bandwidth, you can reduce that resolution and frame rate.
- Based on receiver bandwidth available, determine how many video tiles to show. Don't overshoot 6Mbps for all video tiles and content sharing. End users see black video tiles when they don't have enough bandwidth.

Using video uplink and downlink bandwidth policies

The Amazon Chime SDK provides the following bandwidth policies.

- NScaleVideoUplinkBandwidthPolicy – Implements capture and encoding parameters that nearly equal those used by the desktop, web, and mobile clients.
- AllHighestVideoBandwidthPolicy – Always subscribes to the highest quality video stream.
- NoVideoDownlinkBandwidthPolicy – Disables video when bandwidth drops below a given threshold.
- VideoPriorityBasedPolicy – Prioritizes audio over video in cases of low bandwidth.
- VideoAdaptiveProbePolicy

Verifying SDK quotas and API throttling

The Amazon Chime SDK endpoints and quotas page lists the service quotas, API rates, and whether you can adjust them. Use the AWS Console Service Quota page to request quota adjustments.

Fine tuning your API rates

Applications that exceed their API rates receive HTTP Status Code 429 and `ThrottledClientException` messages. You can adjust your API rates, but before you do, check your application for bugs that may exhaust those rates. For example, you may create meetings in a loop, or create meetings and not clean up.

Depending on how you create meetings, you might need to modify your code. For example, you can replace `CreateMeeting` and `CreateAttendee` with:

- CreateMeetingWithAttendees – Creates up to 10 attendees per meeting.
- BatchCreateAttendee – Creates up to 100 attendees per meeting.

You can store created attendees in a database, pull attendee information as invitees join the meeting, and then associate them with the pre-created attendees.

Opening support cases for Amazon Chime SDK meetings

If you have more questions, or require support for your business, you can reach out to AWS Customer support. For more information about our support plans, see the Compare support plans page. When creating a support case, always open it under the account that has the problem. Include console browser logs, meeting and attendee IDs, and any related support cases or GitHub issues.

Using Amazon Chime SDK messaging

You use this section of the Amazon Chime SDK Developer Guide to help create messaging applications that run on the Amazon Chime SDK service. This SDK provides the conceptual and practical information needed to create a basic messaging app.

Topics

Migrating to the Amazon Chime SDK Identity namespace

The Amazon Chime SDK Identity namespace is a dedicated place for the APIs used to create and manage Amazon Chime SDK identity resources, including AppInstances and AppInstanceUsers. You

use the namespace to address Amazon Chime SDK identity API endpoints in any AWS Region in which they're available. Use this namespace if you're just starting to use the Amazon Chime SDK. For more information about Regions, refer to Available AWS Regions for the Amazon Chime SDK service in this guide.

Existing applications that use the Amazon Chime namespace should plan to migrate to the dedicated namespace.

Topics

Reasons to migrate

We encourage you to migrate to the Amazon Chime SDK Identity namespace for these reasons:

Choice of API Endpoint

The Amazon Chime SDK Identity namespace is the only API namespace that can use API endpoints in any Region that makes them available. If you want to use API endpoints other than `us-east-1`, you must use the Amazon Chime SDK Identity namespace. For more information about the current endpoints, refer to API mapping in this guide.

Updated and new messaging APIs

We only add or update identity APIs in the Amazon Chime SDK Identity namespace.

Before you migrate

Before you migrate, be aware of the differences between the namespaces. The following table lists and describes them.

	Amazon Chime SDK Identity namespace	Amazon Chime namespace
AWS SDK namespace	ChimeSDKIdentity	Chime

	Amazon Chime SDK Identity namespace	Amazon Chime namespace
Regions	Multiple	us-east-1 only
Service principal	https://identity.chime.amazonaws.com	https://chime.amazonaws.com
APIs	Only APIs for identity	APIs for identity and other parts of Amazon Chime
User Expiration	Available	Not available
Bots	Available	Not available

Differences between the namespaces

The following sections explain the differences between the `Chime` and `ChimeSDKIdentity` namespaces.

AWS SDK namespace

The Amazon Chime SDK namespace uses the `Chime` formal name. The Amazon Chime SDK Identity namespace uses the `ChimeSDKIdentity` formal name. The precise format of the name varies by platform.

For example, if you use the AWS SDK in Node.js to create identities, you use a line of code to address the namespace.

```
const chimeIdentity = AWS.Chime();
```

To migrate to the `ChimeSDKIdentity` namespace, update this line of code with the new namespace and the endpoint region.

```
const chimeIdentity = AWS.ChimeSDKIdentity({ region: "eu-central-1" });
```

Regions

The Amazon Chime namespace can only address API endpoints in the `us-east-1` Region. The Amazon Chime SDK Identity namespace can address Amazon Chime SDK Identity API endpoints in any Region they are available. For a current list of endpoints Regions, refer to Available AWS Regions for the Amazon Chime SDK service in this guide.

Endpoints

The Amazon Chime SDK Identity namespace uses different API endpoints than the Amazon Chime namespace.

Only the endpoint used to create identity resources can be used to update them. This means an AppInstance created via an endpoint in `eu-central-1` can only be modified via `eu-central-1`. It also means you cannot address an AppInstance created via the Chime namespace with the ChimeSDKIdentity namespace in `us-east-1`, or create a channel in a Region other than the region where the AppInstance and AppInstanceUser members were created. For more information about the current endpoints, refer to API mapping in this guide.

Service principal

The Amazon Chime SDK Identity namespace uses a new service principal: `Identity.chime.amazonaws.com`. If you have SQS, SNS, or other IAM access policies that grant access to the service, you need to update those polices to grant access to the new service principal.

APIs

The Amazon Chime SDK Identity namespace only contains APIs for creating and managing messaging resources, and for sending and receiving messages. The Amazon Chime namespace includes APIs for other parts of the Amazon Chime service as well as messaging.

User expiration

Expiration settings on creation of AppInstanceUsers allow you to create temporary users. For example, you can create chat users that only exist for the duration of a large broadcast. Only the Identity namespace supports expiration settings for AppInstanceUsers.

Bots

You use the AppInstanceBot API to add chat bots powered by Amazon Lex V2 into your applications. You can only use AppInstanceBots in the Identity namespace. For more information about bots, refer to Using AppInstanceBots as intelligent channel agents in this guide.

Additional APIs

The Identity namespace has a growing list of additional APIs that the Chime namespace does not have. If you are getting started with the Amazon Chime SDK, use the Identity namespace to have access to all of the latest features. For more information about the current APIs, refer to Amazon Chime SDK Identity in the *Amazon Chime SDK API Reference*.

Migrating to the Amazon Chime SDK Messaging namespace

The Amazon Chime SDK Messaging namespace is a dedicated place for the APIs that create and manage Amazon Chime SDK messaging resources. You use the namespace to address Amazon Chime SDK messaging API endpoints in any AWS Region in which they're available. Use this namespace if you're just starting to use the Amazon Chime SDK. For more information about Regions, refer to Available AWS Regions for the Amazon Chime SDK service in this guide.

Existing applications that use the Amazon Chime namespace should plan to migrate to the dedicated namespace.

Topics

Reasons to migrate

We encourage you to migrate to the Amazon Chime SDK Messaging namespace for these reasons:

Choice of API Endpoint

The Amazon Chime SDK Messaging namespace is the only API namespace which can use API endpoints in any Region that makes them available. If you want to use API endpoints other than US East (N. Virginia), you must use the Amazon Chime SDK Messaging namespace.

For more information about how Amazon Chime SDK messaging uses AWS Regions, refer to Available Regions in this guide.

Updated and new messaging APIs

We only add or update messaging APIs in the Amazon Chime SDK Messaging namespace.

Before you migrate

Before you migrate, be aware of the differences between the namespaces. The following table lists and describes them.

	Amazon Chime SDK Messaging namespace	Amazon Chime namespace
AWS SDK namespace	ChimeSDKMessaging	Chime
Regions	Multiple	US East (N. Virginia) only
APIs	Only APIs for messaging	APIs for messaging and other parts of Amazon Chime
Flows	Available	Not available
Elastic channels	Available	Not available

Differences between the namespaces

The following sections explain the differences between the `Amazon Chime` and `Amazon Chime SDK Messaging` namespaces.

AWS SDK namespace

The Amazon Chime SDK namespace uses the `Chime` formal name. The Amazon Chime SDK Messaging namespace uses the `ChimeSDKMessaging` formal name. The precise format of the name varies by platform.

For example, if you use the AWS SDK in Node.js to create messaging, you use a line of code to address the namespace.

```
const chimeMessaging = AWS.Chime();
```

To migrate to the Amazon Chime Messaging SDK, update this line of code with the new namespace and the endpoint region.

```
const chimeMessaging = AWS.ChimeSDKMessaging({ region: "Europe (Frankfurt)" });
```

Regions

The Amazon Chime namespace can only address API endpoints in the `US East (N. Virginia)` Region. The Amazon Chime SDK Messaging namespace can address Amazon Chime SDK messaging API endpoints in any Region they're available. For a current list of messaging Regions, refer to Available AWS Regions for the Amazon Chime SDK service in this guide.

Endpoints

The Amazon Chime SDK Messaging namespace uses different API endpoints than the Amazon Chime namespace.

Only the endpoint used to create a messaging resource can be used to modify it. This means a messaging resource created via an endpoint in `Europe (Frankfurt)` can only be modified via `Europe (Frankfurt)`. This means that a channel created via an endpoint in Europe (Frankfurt) can only be modified via Europe (Frankfurt). It also means that you cannot address a channel created via the `Chime` namespace with the `ChimeSDKMessaging` namespace in US East (N. Virginia). For more information about the current endpoints, refer to API mapping in this guide.

Service principal

The Amazon Chime SDK Messaging namespace uses a new service principal: `messaging.chime.amazonaws.com`. If you have SQS, SNS, or other IAM access policies that grant access to the service, you need to update those polices to grant access to the new service principal.

APIs

The Amazon Chime SDK Messaging namespace only contains APIs for creating and managing messaging resources, and for sending and receiving messages. The Amazon Chime namespace includes APIs for other parts of the Amazon Chime service as well as messaging.

Channel flows

Channel flows allow developers to run business logic on in-flight messages before they are delivered to members of a messaging channel. For example, you can create flows that remove sensitive data such as government ID numbers, phone numbers, or profanity from messages before they are delivered. That can help implement corporate communications policies or other communication guidelines.

You can also use channel flows to perform functions such as aggregating responses to a poll before sending the results back to participants, or sending messages via SMS.

Channel Flows are only available in the `ChimeSDKMessaging` namespace. For more information about them, refer to Using channel flows to process messages in this guide.

Elastic channels

Elastic channels support large scale chat experiences with up to one million chat users automatically balanced across a defined number of sub-channels. Elastic channels are only available in the `ChimeSDKMessaging` endpoint. For more information about elastic channels, refer to Using elastic channels to host live events in this guide.

Additional APIs

The Messaging namespace has a growing list of APIs that the `Chime` namespace does not have. If you are getting started with the Amazon Chime SDK use the messaging namespace for access to all of the latest features. For more information about the current APIs, refer to Amazon Chime SDK Messaging in the *Amazon Chime SDK API Reference*.

Messaging prerequisites

You need the following to use Amazon Chime SDK messaging.

- The ability to program.
- An AWS account.
- Permissions to configure IAM roles for the applications using Amazon Chime SDK messaging.

For the majority of cases, you also need:

- **A client application** – Displays messaging UI, connects to web sockets using the Amazon Chime SDKs, manages state.
- **A server application** – Manages identity and users.

Understanding messaging concepts

To use Amazon Chime SDK messaging effectively, you must understand the following terms and concepts.

AppInstance

To use Amazon Chime SDK messaging, you must first create an AppInstance. An AppInstance contains AppInstanceUsers and Channels. Typically, you create a single AppInstance for your application. An AWS account can have multiple AppInstances. You make app level settings, such as message retention and streaming configuration, at the AppInstance level. AppInstances are identified by a unique ARN in this format: `arn:aws:chime:region:aws_account_id:app-instance/app_instance_id`.

AppInstanceUser

AppInstanceUsers are the entities that send messages, create channels, join channels, and so on. Typically, you create a one-to-one mapping of an `AppInstanceUser` to a user of your app. You can also create an `AppInstanceUser` to connect to back-end services, which allows users to identify messages as coming from a back-end service. AppInstanceUsers identified by an ARN, such as `arn:aws:chime:region:aws_account_id:app-instance/app_instance_id/user/app_instance_user_id`. You control the `app_instance_user_id`, and as a best practice, re-use the IDs that your application already has.

Channel

When you add an `AppInstanceUser` to a channel, that user becomes a member and can send and receive messages. Channels can be public, which allows any user to add themselves as a member, or private, which allows only channel moderators to add members. You can also hide channel members. Hidden members can observe conversations but not send messages, and they aren't added to the channel membership.

SubChannel

Members of an elastic channel are divided into a logical container called SubChannels. When you add an AppInstanceUser to an elastic channel, the user becomes a member of a SubChannel and can send and receive messages for that particular SubChannel. Channel memberships and messages are at a SubChannel level meaning that a message sent by a member in one SubChannel will not be received by a member in another SubChannel. Members are transferred to different SubChannels to support the elastic nature of a channel and promote engagement.

UserMessage

An `AppInstanceUser` who belongs to a channel can send and receive user messages. The `AppInstanceUser` can send `STANDARD` or `CONTROL` messages. `STANDARD` messages can contain

4KB of data and 1KB of metadata. `CONTROL` messages can contain only 30 bytes of data. Messages can be `PERSISTENT` or `NON_PERSISTENT`. You can retrieve `PERSISTENT` messages from the channel history. `NON_PERSISTENT` messages are only seen by channel members currently connected to Amazon Chime SDK messaging.

System Message

Amazon Chime SDK generates system messages in response to events such as members joining or leaving a channel.

Understanding messaging architecture

You can use Amazon Chime SDK messaging as a server-side and a client-side SDK. The server-side APIs create an `AppInstance` and `AppInstanceUser`. You can use various hooks and configurations to add application specific business logic and validation. For more information about doing that, see Streaming messaging data. Additionally, server-side processes can call APIs on behalf of an `AppInstanceUser`, or control a dedicated `AppInstanceUser` that represents back-end processes.

Client-side applications represented as an `AppInstanceUser` can call the Amazon Chime SDK messaging APIs directly. Client-side applications use the WebSocket protocol to connect to the messaging SDK when they are online. When connected, they receive real-time messages from any channel that they are a member of. When disconnected, an `AppInstanceUser` still belongs to the channels it was added to, and it can load the message history of those channels by using the SDK's HTTP based APIs.

Client-side applications have permissions to make API calls as a single `AppInstanceUser`. To scope IAM credentials to a single `AppInstanceUser`, client side applications assume a parameterized IAM role via AWS Cognito Identity Pools, or by a small self-hosted back-end API. For more information about authentication, see Authenticating end-user client applications. In contrast, server side applications typically have permissions tied to a single app instance user, such as a user with administrative permissions, or they have permissions to make API calls on behalf of all app instance users.

Understanding message types

You send messages through channels. You can send `STANDARD`, `CONTROL`, or `SYSTEM` messages.

- `STANDARD` messages can be up to 4KB in size and contain metadata. Metadata is arbitrary, and you can use it in a variety of ways, such as containing a link to an attachment.
- `CONTROL` messages are limited to 30 bytes and do not contain metadata.
- `STANDARD` and `CONTROL` messages can be persistent or non-persistent. Persistent messages are preserved in the history of a channel and viewed by using a `ListChannelMessages` API call. Non persistent messages are sent to every `AppInstanceUser` connected via WebSocket.
- Amazon Chime SDK sends automated `SYSTEM` messages for events such as members joining or leaving a channel.

Getting started

The topics in this section explain how to start building an Amazon Chime SDK messaging application.

Topics

Creating an AppInstance

To use Amazon Chime SDK messaging, you must first create an Amazon Chime SDK `AppInstance` in your AWS account.

Topics

Building an AppInstance

To create an AppInstance for messaging

1. In the CLI, run `aws chime-sdk-identity create-app-instance --name NameOfAppInstance`.
2. In the create response, make note of the AppInstanceArn. `arn:aws:chime:region:aws_account_id:app-instance/app_instance_id`.

Creating an AppInstanceUser

Once you create an AppInstance, you create an AppInstanceUser in that AppInstance. You typically do this when a user first registers or logs in to your app. You can also create an AppInstanceUser that acts on behalf of your back-end services.

The following example shows how to create a back-end AppInstanceUser:

```
aws chime-sdk-identity create-app-instance-user \
    --app-instance-arn "app_instance_arn" \
    --app-instance-user-id "back-end-worker" \
    --name "back-end-worker"
```

In the create response, note the AppInstanceUserArn. It takes this form: `arn:aws:chime:region: aws_account_id:app-instance/app_instance_id/user/app_instance_user_id`. In this example, app_instance_user_id is "back-end-worker."

> **Note**
>
> As a best practice, when creating an AppInstanceUser for a client application, have the AppInstanceUserId match an existing unique ID for that user, such as the sub of an identity provider. The name is an optional placeholder that is attached to some API entities, such as a message sender. It allows you to control the display name of a user in one place, rather then needing to look it up from AppInstanceUser ARN, which is also attached as the sender of a message.

Making SDK calls from a back-end service

Once you create a user to represent your back-end services, you create a channel, send messages to that channel, and read messages from that channel.

Run the following CLI command to create a public channel.

```
aws chime-sdk-messaging create-channel \
    --chime-bearer "app_instance_user_arn" \
    --app-instance-arn "app_instance_arn" \
    --name "firstChannel"
```

The command produces an ARN in this format: `arn:aws:chime:region:aws_account_id:app-instance/app_instance_id/channel/channel_id`.

Topics

- How IAM authorization works for back-end services
- Understanding implicit API authorization
- Sending and listing channel messages

How IAM authorization works for back-end services

In the CLI command from the previous section, note the `chime-bearer` parameter. It identifies the user that creates or interacts with resources such as channels and messages. Nearly all Amazon Chime SDK messaging APIs take `chime-bearer` as a parameter, except APIs meant to be called only by developers, such as `CreateAppInstance`.

The IAM permissions for Amazon Chime SDK messaging APIs require an `app-instance-user-arn` that matches the `chime-bearer` parameter. Additional ARNs—typically channel ARNs—might be required based on the API. For back-end services like the example above, this leads to IAM policies like the following example:

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

```
        "chime:SendChannelMessage",
        "chime:ListChannelMessages",
        "chime:CreateChannelMembership",
        "chime:ListChannelMemberships",
        "chime:DeleteChannelMembership",
        "chime:CreateChannel",
        "chime:ListChannels",
        "chime:DeleteChannel",
        ...
    ],
    "Resource": [
        "arn:aws:chime:region:aws_account_id:app-instance/app_instance_id/user/back-
end-worker",
        "arn:aws:chime:region:aws_account_id:app-instance/app_instance_id/channel/*"
    ]
}
```

Note the `AppInstanceUser` ARN and channel ARN in the `Resource` section. This IAM policy example grants the back-end service permission to make API calls as the user with the ID of "back-end-worker." If you want your back-end service to be able to make calls for the people who use your app, change the `app_instance_user_arn` to `arn:aws:chime:region:aws_account_id:app-instance/app_instance_id/user/*`.

Understanding implicit API authorization

In addition to IAM policies, the Amazon Chime SDK messaging APIs have implicit permissions. For example, an `AppInstanceUser` can only send a message or list a channel membership in channels to which the user belongs. One exception to this is an `AppInstanceUser` who was promoted to `AppInstanceAdmin`. By default, admins have permissions to all the channels in your app. For most use cases, you only need this for back-end services that contain significant business logic.

The following CLI command promotes a back-end user to an admin.

```
aws chime-sdk-identity create-app-instance-admin \
    --app-instance-admin-arn "app_instance_user_arn" \
    --app-instance-arn "app_instance_arn"
```

Sending and listing channel messages

The following CLI command sends channel messages.

```
aws chime-sdk-messaging send-channel-message \
    --chime-bearer "app_instance_user_arn" \
    --channel-arn "channel_arn" \
    --content "hello world" \
    --type STANDARD \
    --persistence PERSISTENT
```

The following CLI commands list channel messages in reverse chronological order.

- `aws chime list-channel-messages`
- `aws chime-sdk-messaging list-channel-messages`

```
aws chime-sdk-messaging list-channel-messages \
    --chime-bearer "app_instance_user_arn" \
    --channel-arn "channel_arn"
```

Authenticating end-user client applications

You can also run Amazon Chime SDK messaging from end-user client applications. Making SDK calls from a back-end service explains how to make API calls such as create-channel, send-channel-message, and list-channel-messages. End user client applications such as browsers and mobile applications make these same API calls. Client applications can also connect via WebSocket to receive real time updates on messages and events to channels they are members of. This section covers how to give IAM credentials to a client application scoped to a specific app instance user. Once the end users have these credentials, they can make the API calls shown in Making SDK calls from a back-end service. To see a full demo of a client application, see https://github.com/aws-samples/amazon-chime-sdk/tree/main/apps/chat. For more information about receiving real-time messages from the channels that a client app belongs to, see Using WebSockets to receive messages.

Providing IAM credentials to end users

Amazon Chime SDK messaging integrates natively with AWS Identity and Access Management (IAM) policies to authenticate incoming requests. The IAM policy defines what an individual user can do. IAM policies can be crafted to provide scoped-down limited credentials for your use case.

For more information on creating policies for Amazon Chime SDK messaging users, see Example IAM roles.

If you have an existing identity provider, you have the following options for integrating your existing identity with Amazon Chime SDK messaging.

- You can use your existing identity provider to authenticate users and then integrate the authentication service with AWS Security Token Service (STS) to create your own credential vending service for clients. STS provides APIs for assuming IAM Roles.
- If you already have a SAML or OpenID compatible identity provider, we recommend using Amazon Cognito Identity Pools, which abstract away calls to AWS STS AssumeRoleWithSAML and AssumeRoleWithWebIdentity. Amazon Cognito integrates with OpenID, SAML, and public identity providers such as Facebook, Login with Amazon, Google, and Sign in with Apple.

If you do not have an identity provider, you can get started with Amazon Cognito User Pools. For an example of how to use Amazon Cognito with the Amazon Chime SDK messaging features, see Build chat features into your application with Amazon Chime SDK messaging.

Alternately, you can use the AWS STS to create your own credential vending service or build your own identity provider.

Using STS to vend credentials

If you already have an IDP such as ActiveDirectory LDAP, and you want to implement a custom credential vending service, or grant access to chat for non-authenticated meeting attendees, you can use the AWS STS AssumeRole API. To do this, you first create an Amazon Chime SDK messaging SDK Role. For more information about creating that role, see Creating a role to delegate permissions to an IAM user .

The IAM role would have permissions to the Amazon Chime SDK messaging action your application would use, similar to the following:

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "chime:GetMessagingSessionEndpoint"
```

```
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "chime:SendChannelMessage",
                "chime:ListChannelMessages",
                "chime:CreateChannelMembership",
                "chime:ListChannelMemberships",
                "chime:DeleteChannelMembership",
                "chime:CreateChannelModerator",
                "chime:ListChannelModerators",
                "chime:DescribeChannelModerator",
                "chime:CreateChannel",
                "chime:DescribeChannel",
                "chime:ListChannels",
                "chime:DeleteChannel",
                "chime:RedactChannelMessage",
                "chime:UpdateChannelMessage",
                "chime:Connect",
                "chime:ListChannelBans",
                "chime:CreateChannelBan",
                "chime:DeleteChannelBan",
                "chime:ListChannelMembershipsForAppInstanceUser"
                "chime:AssociateChannelFlow",
                "chime:DisassociateChannelFlow",
                "chime:GetChannelMessageStatus"
            ],
            "Resource": [
                "{chime_app_instance_arn}/user/
${aws:PrincipalTag/my_applications_user_id}",
                "{chime_app_instance_arn}/channel/*"
            ]
        }
    ]
}
```

For this example, call this role the *ChimeMessagingSampleAppUserRole*.

Note the session tag in the *ChimeMessagingSampleAppUserRole* policy `${my_application_user_id}` in the user ARN resource. This session tag is parameterized in the AssumeRole API call to limit the credentials returned to permissions for a single user.

The AssumeRole and TagSession APIs are called using an already credentialed IAM entity, such as an IAM user. The APIs can also be called by a different IAM role such as an AWS Lambda run role. That IAM identity must have permissions to call `AssumeRole` and `TagSession` on the *ChimeMessagingSampleAppUserRole*.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
                ],
            "Resource":
 "arn:aws:iam::my_aws_account_id:role/ChimeMessagingSampleAppUserRole"
        }
    ]
}
```

For this example, call this role the *ChimeSampleAppServerRole*.

You need to set up the `ChimeMessagingSampleAppUserRole` with a trust policy that allows the `ChimeMessagingSampleAppServerRole` to call the STS AssumeRole API on it. For more information about using trust policies with IAM roles, see How to use trust policies with IAM roles . You can use the AWS IAM Roles Console to add this policy to the `ChimeMessagingSampleAppUserRole`. The following example shows a typical trust relationship.

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

 "AWS":"arn:aws:iam::my_aws_account_id:role/ChimeMessagingSampleAppServerRole"
            }
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
```

In a sample deployment, an Amazon EC2 instance, or an AWS Lambda is launched with the `ChimeMessagingSampleAppServerRole`. The server then:

1. Performs any application specific authorization on a client's requests to receive credentials.
2. Calls STS `AssumeRole` on `ChimeMessagingSampleAppUserRole`, with a tag parameterizing the `${aws:PrincipalTag/my_applications_user_id}`.
3. Forwards the credentials returned in the `AssumeRole` call to the user.

The following example shows CLI command for assuming a role for step 2:

```
aws sts assume-role --role-arn
arn:aws:iam::my_aws_account_id:role/ChimeMessagingSampleAppUserRole --role-
session-name demo --tags Key=my_applications_user_id,Value=123456789
```

Creating channels

You and your end users can create channels. Once created, you or your end users also need to add members to the channel. Sample code for creating channels is available in the sample application on GitHub.

For more information on creating channels and adding members, see:

- CreateChannel
- CreateChannelMembership

Sending messages

Use the SendChannelMessage API to send messages. Sample code is available in a sample application on GitHub.

Using ExpirationSettings

When you create an `AppInstanceUser` or a `Channel`, you can use `ExpirationSettings` to configure those resources for automatic deletion. `ExpirationSettings` helps reduce storage

costs and prevent resource-limit-exceeded issues. For example, you can delete unused channels after 7 days, or delete an `AppInstanceUser` that was only invoked for testing purposes.

For an `AppInstanceUser`, you specify the expiration period based on user creation time. For a `Channel`, you specify the expiration period based on the channel's creation time, or last message time. The latter allows you use message activities to customize automatic deletion.

> **Important**
>
> Shortly after a resource expires, `ExpirationSettings` starts a background process to delete that resource. The process usually takes 6 hours, but that time can vary. Expired `AppInstanceUsers` and `Channels` that haven't yet been deleted still appear as valid and active. You can update or remove their expiration settings, and the system honors your changes.

Topics

Configuring ExpirationSettings

The following sections explain how to configure the `ExpirationSettings` of an `AppInstanceUser` or a `Channel`.

Configuring ExpirationSettings when you create a resource

You can configure `ExpirationSettings` when you run the CreateAppInstanceUser or CreateChannel APIs. If you set the `ExpirationSettings` parameter, you must grant the following IAM permissions:

- `chime:PutAppInstanceUserExpirationSettings` when creating an `AppInstanceUser`
- `chime:PutChannelExpirationSettings` when creating a `Channel`.

The following example uses the AWS CLI to create an `AppInstanceUser` that expires after a day.

```
aws chime-sdk-identity create-app-instance-user \
```

```
--app-instance-arn "app_instance_arn" \
--app-instance-user-id "backend-worker" \
--name "backend-worker" \
--expiration-settings '{
            "ExpirationDays": 1,
            "ExpirationCriterion": "CREATED_TIMESTAMP"
        }'
```

The following example uses the AWS CLI to create a `Channel` that expires after a day after it last receives a message.

```
aws chime-sdk-messaging create-channel \
--chime-bearer "app_instance_user_arn" \
--app-instance-arn "app_instance_arn" \
--name "firstChannel" \
--expiration-settings '{
            "ExpirationDays": 1,
            "ExpirationCriterion": "LAST_MESSAGE_TIMESTAMP"
        }'
```

Using Put APIs to configure ExpirationSettings

You can also use the PutAppInstanceUserExpirationSettings and PutChannelExpirationSettings APIs to create, update, and delete `ExpirationSettings`.

The following example shows you to use the AWS CLI to update an `AppInstanceUser`'s `ExpirationSettings`.

```
aws chime-sdk-identity put-app-instance-user-expiration-settings \
--app-instance-user-arn "app_instance_user_arn" \
--expiration-settings '{
            "ExpirationDays": 30,
            "ExpirationCriterion": "CREATED_TIMESTAMP"
        }'
```

The following example shows you to use the AWS CLI to delete a channel's `ExpirationSettings`.

```
aws chime-sdk-messaging put-channel-expiration-settings \
--chime-bearer "app_instance_user_arn" \
--channel-arn "channel_arn"
```

AWS CloudTrail events for expired resource deletion

After the system deletes an expired resource, it sends an `ExpireAppInstanceUser` or `ExpireChannel` event to AWS CloudTrail. The type of event depends on the type of deleted asset.

The following example shows an `AppInstanceUser` event.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "accountId": "123456789012",
        "invokedBy": "chime.amazonaws.com"
    },
    "eventTime": "2023-03-15T00:00:00Z",
    "eventSource": "chime.amazonaws.com",
    "eventName": "ExpireAppInstanceUser",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "chime.amazonaws.com",
    "userAgent": "chime.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "12345678-1234-1234-1234-123456789012",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::Chime::AppInstanceUser",
            "ARN": "arn:aws:chime:us-east-1:123456789012:app-instance/app-instance-id/
user/user-id"
        }
    ],
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails": {
        "reason": "AppInstanceUser deleted due to expiration settings."
    },
    "eventCategory": "Management"
}
```

Using WebSockets to receive messages

You can use the Amazon Chime JS SDK to receive messages using WebSockets, or you can use the WebSocket client library of your choice.

Follow these topics in the order listed to start using WebSockets:

Topics

Defining an IAM policy

To start, define an IAM policy that gives you permission to establish a WebSocket connection. The following example policy gives an `AppInstanceUser` permission to establish a WebSocket connection.

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action: [
      "chime:Connect"
    ],
    "Resource": [
      "arn:aws:chime:region:{aws_account_id}:app-instance/{app_instance_id}/user/
{app_instance_user_id}"
    ]
 },
 {
    "Effect": "Allow",
    "Action: [
      "chime:GetMessagingSessionEndpoint"
    ],
    "Resource": [
      "*"
```

```
    ]
  }
  ]
}
```

Retrieving the endpoint

The following steps explain how to retrieve the endpoint used in a WebSocket connection.

1. Use the GetMessagingSessionEndpoint API to retrieve the WebSocket endpoint.
2. Use the URL returned by the GetMessagingSessionEndpoint API to construct a Signature Version 4 Signed WebSocket URL. If you need help doing that, you can follow directions in the Establishing the connection.

> **Note**
>
> WebSocket URLs have the following form: `id.region.ws-messaging.chime.aws`

Establishing the connection

After you retrieve an endpoint, you use the connect API to establish a WebSocket connection to the Amazon Chime SDK back-end server and receive messages for an `AppInstanceUser`. You must use AWS Signature Version 4 to sign requests. For more information about signing a request, see Signing AWS Requests with Signature Version 4.

> **Note**
>
> To retrieve the endpoint, you can invoke the GetMessagingSessionEndpoint API. You can use the WebSocket client library of your choice to connect to the endpoint.

Request Syntax

```
GET /connect
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIARALLEXAMPLE%2F20201214%2Fregion%2Fchime%2Faws4_request
&X-Amz-Date=20201214T171359Z
```

```
&X-Amz-Expires=10
&X-Amz-SignedHeaders=host
&sessionId={sessionId}
&userArn={appInstanceUserArn}
&X-Amz-Signature=db75397d79583EXAMPLE
```

URI Request Parameters

All URI Request Query Parameters must be URL Encoded.

X-Amz-Algorithm

Identifies the version of AWS Signature and the algorithm that you used to calculate the signature. The Amazon Chime SDK supports only AWS Signature Version 4 authentication, so the value of this is `AWS4-HMAC-SHA256`.

X-Amz-Credential

In addition to your access key ID, this parameter also provides the AWS Region and service—the scope—for which the signature is valid. This value must match the scope you use in signature calculations. The general form for this parameter value is:

```
<yourAccessKeyId>/<date>/<awsRegion>/<awsService >/aws4_request
```

For example:

```
AKIAIOSFODNN7EXAMPLE/20201214/us-east-1/chime/aws4_request
```

X-Amz-Date

The date and time format must follow the ISO 8601 standard, and you must format it as `yyyyMMddTHHmmssZ`. For example, you must convert **08/01/2020 15:32:41.982-700** to Coordinated Universal Time (UTC) and submit it as `20200801T083241Z`.

X-Amz-Signed-Headers

Lists the headers that you used to calculate the signature. The following headers are required in the signature calculations:

- The HTTP host header.
- Any x-amz-* headers that you plan to add to the request.

> **Note**
>
> For added security, sign all the request headers that you plan to include in your request.

X-Amz-Signatures

Provides the signature to authenticate your request. This signature must match the signature that Amazon Chime SDK calculates. If it doesn't, Amazon Chime SDK denies the request. For example, `733255ef022bec3f2a8701cd61d4b371f3f28c9f19EXAMPLEd48d5193d7`.

X-Amz-Security-Token

Optional credential parameter if using credentials sourced from the Security Token Service. For more information about the service, see the https://docs.aws.amazon.com/STS/latest/APIReference/.

SessionId

Indicates a unique Id for the WebSocket connection being established.

UserArn

Indicates the identity of the `AppInstanceUser` trying to establish a connection. The value should be the ARN of the `AppInstanceUser`. For example, `arn:aws:chime:us%2Deast%2D1:123456789012:app%2Dinstance/694d2099%2Dcb1e%2D463e%2D9d64%2D697ff5b8950e/user/johndoe`

Using prefetch to deliver channel details

When you establish a WebSocket connection, you can specify `prefetch-on=connect` in your query parameters to deliver `CHANNEL_DETAILS` events. The prefetch feature comes with the connect API, and the feature enables users to see an enriched chat view without extra API calls. Users can:

- See a preview of the last channel message, plus its timestamp.
- See the members of a channel.
- See a channel's unread markers.

After a user connects with the prefetch parameter specified, the user receives the session established event, which indicates the connection has been established. The user then receives up to 50 `CHANNEL_DETAILS` events. If the user has less than 50 channels, the connect API prefetches all channels via `CHANNEL_DETAILS` events. If user has more than 50 channels, the API prefetches the top 50 channels that contain unread messages and the latest `LastMessageTimestamp` values. The `CHANNEL_DETAILS` events arrive in random order, and you receive events for all 50 channels.

Also, prefetch returns the following for `ChannelMessages` and `ChannelMemberships`:

- **ChannelMessages** – List of ChannelMessageSummary objects, ordered by `CreatedTimestamp` in descending order. Only includes the latest 20 messages visible to the user. If there are targeted messages in the channel that are not visible to the current user, then less than 20 messages might be returned. The `ChannelMessagesHasMore` boolean will be set to true to indicate there are more messages. Soft limit, adjustable at the AWS account level.
- **ChannelMemberships** – List of ChannelMembershipSummary objects. Includes a maximum of 30 channel members. Soft limit, adjustable at AWS account level.

This example shows how to use `prefetch-on=connect`.

```
GET /connect
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIARALLEXAMPLE%2F20201214%2Fregion%2Fchime%2Faws4_request
&X-Amz-Date=20201214T171359Z
&X-Amz-Expires=10
&X-Amz-SignedHeaders=host
&sessionId=sessionId
&prefetch-on=connect
&userArn=appInstanceUserArn
&X-Amz-Signature=db75397d79583EXAMPLE
```

This example shows the response for one channel. You will receive responses for all 50 channels.

```
{
   "Headers": {
        "x-amz-chime-event-type": "CHANNEL_DETAILS",
        "x-amz-chime-message-type": "SYSTEM"
        },
   "Payload": JSON.stringify"({
        Channel: ChannelSummary
        ChannelMessages: List of ChannelMessageSummary
```

```
        ChannelMemberships: List of ChannelMembershipSummary
        ReadMarkerTimestamp: Timestamp
        ChannelMessagesHasMore: Boolean
    })
}
```

Processing the events

For an `AppInstanceUser` to receive messages after they establish a connection, you must add them to a channel. To do that, use the CreateChannelMembership API.

> **Note**
>
> An `AppInstanceUser` always receives messages for all channels that they belong to. Messaging stops when the `AppInstance` user disconnects.

An `AppInstanceAdmin` and a `ChannelModerator` do not receive messages on a channel unless you use the CreateChannelMembership API to explicitly add them.

The following topics explain how to process events.

Topics

- Understanding message structures
- Handling disconnects

Understanding message structures

Every WebSocket message that you receive adheres to this format:

```
{
   "Headers": {"key": "value"},
   "Payload": "{\"key\": \"value\"}"
}
```

Headers

Amazon Chime SDK messaging use the following header keys:

- `x-amz-chime-event-type`

- `x-amz-chime-message-type`
- `x-amz-chime-event-reason`

The next section lists and describes the header's possible values and payloads.

Payload

Websocket messages return JSON strings. The structure of the JSON strings depends on the `x-amz-event-type` headers. The following table lists the possible `x-amz-chime-event-type` values and payloads:

EventType	Payload format	
`SESSION_ESTABLISHED`	N/A. This message is sent once after the user connects to the WebSocket. It indicates that any message or event on a channel that arrives after the user receives the `SESSION_ESTABLISHED` message is guaranteed to be delivered to the user as long as the WebSocket stays open.	
`CREATE_CHANNEL_MESSAGE`	ChannelMessage	
`REDACT_CHANNEL_MESSAGE`		
`UPDATE_CHANNEL_MESSAGE`		
`DELETE_CHANNEL_MESSAGE`		
`PENDING_CREATE_CHANNEL_MESSAGE`		

EventType	Payload format	
PENDING_UPDATE_CHANNEL_MESSAGE		
FAILED_CREATE_CHANNEL_MESSAGE		
FAILED_UPDATE_CHANNEL_MESSAGE		
DENIED_CREATE_CHANNEL_MESSAGE		
DENIED_UPDATE_CHANNEL_MESSAGE		

EventType	Payload format	
CHANNEL_DETAILS	**Channel** The ChannelSummary object. **ChannelMessages** List of ChannelMessageSummary objects, ordered by `CreatedTimestamp` in descending order. Includes the latest 20 messages, but you can adjust that limit at the AWS account level. **ChannelMemberships** List of ChannelMembershipSummary objects. Returns a maximum of 30 channel members, but you can adjust that limit at the AWS account level. **ReadMarkerTimestamp** The time at which the `AppInstanceUser` last marked the channel as read.	
UPDATE_CHANNEL	Channel	
DELETE_CHANNEL		
BATCH_CREATE_CHANNEL_MEMBERSHIP	BatchChannelMembership	

EventType	Payload format	
CREATE_CHANNEL_MEMBERSHIP	ChannelMembership	
DELETE_CHANNEL_MEMBERSHIP		
UPDATE_CHANNEL_MEMBERSHIP		

x-amz-chime-message-type

The following table lists the `x-amz-chime-message-type` message types .

Message type	Description
STANDARD	Sent when the websocket receives a STANDARD channel message.
CONTROL	Sent when the WebSocket receives a CONTROL channel message.
SYSTEM	All other websocket messages sent by Amazon Chime SDK Messaging.

x-amz-chime-event-reason

This is an optional header supported for a specific use case. The header provides information about why a specific event was received.

Event reason	Description
subchannel_DELETED	DELETE_CHANNEL_MEMBERSHIP events received by elastic channel moderators. Only seen by moderators after membershi

Event reason	Description
	p balancing deletes a sub-channel that they belonged to.

Handling disconnects

Websockets can disconnect due to changes in network connectivity, or when credentials expire. After you open a WebSocket, the Amazon Chime SDK sends regular pings to the messaging client to ensure it is still connected. If the connection closes, the client receives a WebSocket close code. The client can try to reconnect, or not, depending on the close code. The following tables show the close codes that the client can use to reconnect.

For 1000 to 4000 closure codes, reconnect only for the following messages:

Closure codes	Can reconnect	Reason
1001	Yes	Normal closure
1006	Yes	Abnormal closure
1011	Yes	Internal server error
1012	Yes	Service restart
1013	Yes	Try again later
1014	Yes	The server was acting as a gateway or proxy and received an invalid response from the upstream server. This is similar to the 502 HTTP Status Code.

For 4XXX codes, always reconnect *except* for the following messages:

Closure codes	Can reconnect	Reason
4002	No	Client initiated
4003	No	Forbidden
4401	No	Not authorized

When the application uses a close code to reconnect, the application should:

1. Call the GetMessagingSessionEndpoint API again to obtain a new base URL.
2. Refresh the IAM credentials if they've expired.
3. Connect via the WebSocket.

If you use the amazon-chime-sdk-js library, this is handled for you if you implement the needsRefresh() property and the refresh() method. For a working example, see https://github.com/aws-samples/amazon-chime-sdk/blob/dc11c4c76c78d28f618577706bba2087919a5635/apps/chat/src/providers/AuthProvider.jsx#L93-L101.

Configuring attachments

The Amazon Chime SDK allows you to use your own storage for message attachments, and include them as message metadata. Amazon Simple Storage Service (S3) is the easiest way to get started with attachments.

To use S3 for attachments

1. Create an S3 bucket to store attachments.
2. Create an IAM policy for the bucket that allows Amazon Chime SDK users to upload, download, and delete attachments from your S3 bucket.
3. Create an IAM role for use by your Identity provider to vend credentials to users for attachments.

The sample application provides an example of how to do this with Amazon S3, Amazon Cognito, and the Amazon Chime SDK.

Understanding system messages

Amazon Chime SDK sends system messages to all connected clients for events that take place in channels. Events include:

- `UPDATE_CHANNEL` – This event signifies any update made to the channel details, such as the name or metadata.
- `DELETE_CHANNEL` – This event signifies that the channel and all of its data, including messages, memberships, moderators and bans, will be deleted.
- `CREATE_CHANNEL_MEMBERSHIP` – This event signifies that a particular `AppInstanceUser` has been added as a member to the channel. The event also contains details of the new `AppInstanceUser`.
- `DELETE_CHANNEL_MEMBERSHIP` – This event signifies that an `AppInstanceUser` has been removed from the channel. The event also contains the removed `AppInstanceUser` details.
- `UPDATE_CHANNEL_MEMBERSHIP` – This event only applies to elastic channels. The event signifies that membership balancing transferred an `AppInstanceUser` from one sub-channel to another. The event also contains the `AppInstanceUser` details, plus the information about the sub-channel that the `AppInstanceUser` was transferred to.

Example IAM roles

For users to access the Amazon Chime SDK messaging features, you must define an IAM role and policy to provide credentials to users when they sign in. The IAM policy defines the resources that users can access.

The examples in this section provide basic policies that you can adapt to suit your needs. For more information about how policies work, see Making SDK calls from a back-end service.

This example shows a policy for developers building applications using Amazon Chime SDK messaging.

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

```
                "chime:DescribeAppInstance",
                "chime:ListAppInstances",
                "chime:UpdateAppInstance",
                "chime:DeleteAppInstance",
                "chime:CreateAppInstanceUser",
                "chime:DeleteAppInstanceUser",
                "chime:ListAppInstanceUsers",
                "chime:UpdateAppInstanceUser",
                "chime:DescribeAppInstanceUser",
                "chime:CreateAppInstanceAdmin",
                "chime:DescribeAppInstanceAdmin",
                "chime:ListAppInstanceAdmins",
                "chime:DeleteAppInstanceAdmin",
                "chime:PutAppInstanceRetentionSettings",
                "chime:GetAppInstanceRetentionSettings",
                "chime:PutAppInstanceStreamingConfigurations",
                "chime:GetAppInstanceStreamingConfigurations",
                "chime:DeleteAppInstanceStreamingConfigurations",
                "chime:TagResource",
                "chime:UntagResource",
                "chime:ListTagsForResource"
                "chime:CreateChannelFlow",
                "chime:UpdateChannelFlow",
                "chime:DescribeChannelFlow",
                "chime:DeleteChannelFlow",
                "chime:ListChannelFlows",
                "chime:ListChannelsAssociatedWithChannelFlow",
                "chime:ChannelFlowCallback",
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

This example shows a policy that allows users to access the Amazon Chime SDK user actions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "chime:GetMessagingSessionEndpoint",
            "Effect": "Allow",
```

```
            "Resource": "*"
        },
        {
            "Action": [
                "chime:CreateChannel",
                "chime:DescribeChannel",
                "chime:DeleteChannel",
                "chime:UpdateChannel",
                "chime:ListChannels",
                "chime:Listsubchannels",
                "chime:ListChannelMembershipsForAppInstanceUser",
                "chime:DescribeChannelMembershipForAppInstanceUser",
                "chime:ListChannelsModeratedByAppInstanceUser",
                "chime:DescribeChannelModeratedByAppInstanceUser",
                "chime:UpdateChannelReadMarker",
                "chime:CreateChannelModerator",
                "chime:DescribeChannelModerator",
                "chime:ListChannelModerators",
                "chime:DeleteChannelModerator",
                "chime:SendChannelMessage",
                "chime:GetChannelMessage",
                "chime:DeleteChannelMessage",
                "chime:UpdateChannelMessage",
                "chime:RedactChannelMessage",
                "chime:ListChannelMessages",
                "chime:CreateChannelMembership",
                "chime:DescribeChannelMembership",
                "chime:DeleteChannelMembership",
                "chime:ListChannelMemberships",
                "chime:CreateChannelBan",
                "chime:DeleteChannelBan",
                "chime:ListChannelBans",
                "chime:DescribeChannelBan",
                "chime:Connect"
                "chime:AssociateChannelFlow",
                "chime:DisassociateChannelFlow",
                "chime:GetChannelMessageStatus"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:chime:region:{aws_account_id}:app-instance/{app_instance_id}/
user/{app_instance_user_id}",
                "arn:aws:chime:region:{aws_account_id}:app-instance/{app_instance_id}/
channel/*"
```

```
            ]
        }
    ]
}
```

This example shows a policy that gives users minimal access to Amazon Chime SDK user actions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "chime:GetMessagingSessionEndpoint",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "chime:ListChannels",
                "chime:DescribeChannel",
                "chime:ListChannelMembershipsForAppInstanceUser",
                "chime:DescribeChannelMembershipForAppInstanceUser",
                "chime:ListChannelsModeratedByAppInstanceUser",
                "chime:DescribeChannelModeratedByAppInstanceUser",
                "chime:SendChannelMessage",
                "chime:GetChannelMessage",
                "chime:ListChannelMessages",
                "chime:Connect"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:chime:region:{aws_account_id}:app-instance/{app_instance_id}/
user/{app_instance_user_id}",
                "arn:aws:chime:region:{aws_account_id}:app-instance/{app_instance_id}/
channel/*"
            ]
        }
    ]
}
```

This example shows a policy for establishing a WebSocket connection for an `AppInstanceUser`. For more information about WebSocket connections, see Using WebSockets to receive messages.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
   {
     "Effect": "Allow",
     "Action": [
             "chime:Connect"
            ],
     "Resource": [
             "arn:aws:chime:region:{aws_account_id}:app-instance/{app_instance_id}/
user/{app_instance_user_id}"
         ]
      }
   ]
}
```

Understanding authorization by role

The tables in this topic list the actions that app instance users can run, depending on their role.

Legend

- **Allowed** – If the correct Action/Resource context is specified in the IAM Policy, then it can be successfully run.
- **Allowed with restrictions** – If correct Action/Resource context is specified in the IAM Policy then certain conditions have to be met to successfully run the action.
- **Denied** – Even if correct Action/Resource context is specified in the IAM Policy, it will still be blocked by the back end.

Topics

AppInstanceAdmin

App instance administrators can perform actions on a channels within the app instance for which they are an admin.

API name	Allowed or denied	Notes
`UpdateChannel`	Allowed with restriction	Cannot update ElasticChannelConfiguration once set
`DeleteChannel`	Allowed	
`DescribeChannel`	Allowed	
`ListChannel`	Allowed	
`ListChannelMembershipsForAppInstanceUser`	Allowed	You can also populate the AppInstanceUserArn with another `AppInstanceUser` .
`DescribeChannelMembershipForAppInstanceUser`	Allowed	You can also populate AppInstanceUserArn with another `AppInstanceUser` .
`ListChannelsModeratedByAppInstanceUser`	Allowed	You can also populate AppInstanceUserArn with another AppInstanceUser.
`DescribeChannelModeratedByAppInstanceUser`	Allowed	You can also populate AppInstanceUserArn with another `ppInstanceUser` A. No allowed for elastic channels.
`CreateChannelMembership`	Allowed	
`DescribeChannelMembership`	Allowed	
`ListChannelMembership`	Allowed	

API name	Allowed or denied	Notes
`DeleteChannelMembership`	Allowed	
`SendChannelMessage`	Allowed with restriction	You first need to use CreateChannelMembership to create a membership for yourself, and then call the API.
`GetChannelMessage`	Allowed	
`ListChannelMessage`	Allowed	
`DeleteChannelMessage`	Allowed	
`RedactChannelMessage`	Allowed	
`UpdateChannelMessage`	Allowed with restriction	You can only edit your own messages.
`CreateChannelModerator`	Allowed	
`DeleteChannelModerator`	Allowed	
`DescribeChannelModerator`	Allowed	
`ListChannelModerator`	Allowed	
`CreateChannelBan`	Allowed with restriction	The `AppInstanceUser` that you ban cannot be an `AppInstanceAdmin` or the moderator of that channel.
`DeleteChannelBan`	Allowed with restriction	

API name	Allowed or denied	Notes
`DescribeChannelBan`	Allowed	
`ListChannelBan`	Allowed	
`UpdateChannelReadMarker`	Allowed with restriction	For non-elastic channels, You need to use the CreateChannelMembership API to create a membership for yourself first, and then call the API. Not allowed for elastic channels.
`GetChannelMessage`	Allowed with Restriction	Allowed only for sent messages. Not allowed for messages in processing by channel flow unless you are the message sender.
`ListChannelMessages`	Allowed	
`DeleteChannelMessage`	Allowed with Restriction	Allowed only for sent messages.
`RedactChannelMessage`	Allowed with Restriction	Allowed only for sent messages.
`UpdateChannelMessage`	Allowed with Restriction	You can only edit your own sent messages.
`AssociateChannelFlow`	Allowed	
`DisassociateChannelFlow`	Allowed	
`GetChannelMessageStatus`	Allowed with Restriction	You can only get message status for your own messages.

API name	Allowed or denied	Notes
`ListSubChannels`	Allowed	

ChannelModerator

Channel moderators can perform actions only on channels for which they have the moderator role.

> **Note**
>
> A moderator who is an `AppInstanceAdmin` can perform actions on channels allowed by that role.

API name	Allowed or denied	Notes
`UpdateChannel`	Allowed	Cannot update ElasticChannelConfiguration once set
`DeleteChannel`	Allowed	
`DescribeChannel`	Allowed with restriction	You can only get details for public channels.
`ListChannel`	Allowed with restriction	You can only get details for public channels.
`ListChannelMembershipsForAppInstanceUser`	Allowed with restriction	You can only use your ARN as the AppInstanceUserArn value.
`DescribeChannelMembershipForAppInstanceUser`	Allowed with restriction	You can only use your ARN as the AppInstanceUserArn value.
`ListChannelsModeratedByAppInstanceUser`	Allowed with restriction	You can only use your ARN as the AppInstanceUserArn value.

API name	Allowed or denied	Notes
`DescribeChannelModeratedByAppInstanceUser`	Allowed with restriction	You can also populate an AppInstanceUserArn with another AppInstanceUser.
`CreateChannelMembership`	Allowed	
`DescribeChannelMembership`	Allowed	
`ListChannelMembership`	Allowed	
`DeleteChannelMembership`	Allowed	
`SendChannelMessage`	Allowed with restriction	You need to use CreateChannelMembership API to create a membership for yourself first, and then call the `SendChannelMessage` API.
`GetChannelMessage`	Allowed	
`ListChannelMessage`	Allowed	
`DeleteChannelMessage`	Denied	
`RedactChannelMessage`	Allowed	
`UpdateChannelMessage`	Allowed with restriction	You can only update your own messages.

API name	Allowed or denied	Notes
`CreateChannelModerator`	Allowed	You need to use the CreateChannelMembership API to create a membership for yourself first, and then call the `CreateChannelModerator` API.
`DeleteChannelModerator`	Allowed	
`DescribeChannelModerator`	Allowed	
`ListChannelModerator`	Allowed	
`CreateChannelBan`	Allowed with restriction	The `AppInstanceUser` you are banning cannot be an `AppInstanceAdmin` or the moderator of that channel.
`DeleteChannelBan`	Allowed with restriction	
`DescribeChannelBan`	Allowed	
`ListChannelBan`	Allowed	
`UpdateChannelReadMarker`	Allowed with restriction	For non-elastic channels, you need to use CreateChannelMembership to create a membership for yourself first, and then call the `UpdateChannelReadMarker` API. Not allowed for elastic channels.

API name	Allowed or denied	Notes
`GetChannelMessage`	Allowed with Restriction	Allowed only for sent messages. Not allowed for messages in processing by channel flow unless you are the message sender.
`ListChannelMessages`	Allowed	
`DeleteChannelMessage`	Denied	
`RedactChannelMessage`	Allowed with Restriction	Allowed only for sent messages.
`UpdateChannelMessage`	Allowed with Restriction	You can only edit your own sent messages.
`AssociateChannelFlow`	Allowed	
`DisassociateChannelFlow`	Allowed	
`GetChannelMessageStatus`	Allowed with Restriction	You can only get message status for your own messages.
`ListSubChannels`	Allowed	

Member

An `AppInstanceUser` becomes a member of a channel if they are added to the channel via the CreateChannelMembership API.

Members can perform actions only on channels to which they belong.

> **Note**
>
> A member who is an `AppInstanceAdmin` or `ChannelModerator` can perform actions on Channels allowed by those two roles.

API name	Allowed or denied	Notes
`UpdateChannel`	Denied	
`DeleteChannel`	Denied	
`DescribeChannel`	Allowed with restriction	You can only get details for public channels.
`ListChannel`	Allowed with restriction	You can only get details for public channels.
`ListChannelMembershipsForAppInstanceUser`	Allowed with restriction	You can only use your ARN as the AppInstanceUserArn value.
`DescribeChannelMembershipForAppInstanceUser`	Allowed with restriction	You can only use your ARN as the AppInstanceUserArn value.
`ListChannelsModeratedByAppInstanceUser`	Allowed with restriction	You can only use your ARN as the AppInstanceUserArn value.
`DescribeChannelModeratedByAppInstanceUser`	Allowed with restriction	You can also populate an AppInstanceUserArnwith another AppInstanceUser. Not allowed for elastic channels.

API name	Allowed or denied	Notes
`CreateChannelMembership`	Allowed with restriction	You can only add other members for an UNRESTRICTED channel.
`DescribeChannelMembership`	Allowed	
`ListChannelMembership`	Allowed	
`DeleteChannelMembership`	Allowed	
`SendChannelMessage`	Allowed	
`GetChannelMessage`	Allowed	
`ListChannelMessage`	Allowed	
`DeleteChannelMessage`	Denied	
`RedactChannelMessage`	Allowed with restriction	You can only redact your own messages.
`UpdateChannelMessage`	Allowed with restriction	You can only update your own messages.
`CreateChannelModerator`	Denied	
`DeleteChannelModerator`	Denied	
`DescribeChannelModerator`	Denied	
`ListChannelModerator`	Denied	

API name	Allowed or denied	Notes
`CreateChannelBan`	Denied	
`DeleteChannelBan`	Denied	
`DescribeChannelBan`	Denied	
`ListChannelBan`	Denied	
`UpdateChannelReadMarker`	Allowed with restriction	Not allowed for elastic channels.
`GetChannelMessage`	Allowed with Restriction	Allowed only for sent messages. Not allowed for messages in processing by channel flow unless you are the message sender.
`ListChannelMessages`	Allowed	
`DeleteChannelMessage`	Allowed with Restriction	Allowed only for sent messages.
`RedactChannelMessage`	Allowed with Restriction	Allowed only for sent messages.
`UpdateChannelMessage`	Allowed with Restriction	You can only edit your own sent messages.
`AssociateChannelFlow`	Denied	
`DisassociateChannelFlow`	Denied	
`GetChannelMessageStatus`	Allowed with Restriction	You can only get message status for your own messages.
`Listsubchannels`	Denied	

Non-member

Non-members are a regular `AppInstanceUser` and they cannot perform any channel related actions unless you use the CreateChannelMembership API to add them.

> **Note**
>
> A non-member who is an `AppInstanceAdmin` or `ChannelModerator` can perform channel related actions allowed by those two roles.

API name	Allowed or denied	Notes
`UpdateChannel`	Denied	
`DeleteChannel`	Denied	
`DescribeChannel`	Allowed with restriction	You can only get details for public channels.
`ListChannel`	Allowed with restriction	You can only get details for public channels.
`ListChannelMembershipsForAppInstanceUser`	Allowed with restriction	You can only use your ARN as the AppInstanceUserArn value.
`DescribeChannelMembershipForAppInstanceUser`	Allowed with restriction	You can also populate an AppInstanceArn with another `AppInstanceUser` . Not allowed for elastic channels.
`ListChannelsModeratedByAppInstanceUser`	Allowed with restriction	You can only use your ARN as the AppInstanceUserArn value.

API name	Allowed or denied	Notes
`DescribeChannelModeratedByAppInstanceUser`	Allowed with restriction	You can only use your ARN as the AppInstanceUserArn value.
`CreateChannelMembership`	Denied	
`DescribeChannelMembership`	Allowed with restriction	You can only get details for public channels.
`ListChannelMembership`	Allowed with restriction	You can only get details for public channels.
`DeleteChannelMembership`	Denied	
`SendChannelMessage`	Denied	
`GetChannelMessage`	Allowed with restriction	You can only get details for public channels.
`ListChannelMessage`	Allowed with restriction	You can only get details for public channels.
`DeleteChannelMessage`	Denied	
`RedactChannelMessage`	Denied	
`UpdateChannelMessage`	Denied	
`CreateChannelModerator`	Denied	
`DeleteChannelModerator`	Denied	
`DescribeChannelModerator`	Denied	

API name	Allowed or denied	Notes
`ListChannelModerator`	Denied	
`CreateChannelBan`	Denied	
`DeleteChannelBan`	Denied	
`DescribeChannelBan`	Denied	
`ListChannelBan`	Denied	
`UpdateChannelReadMarker`	Denied	
`GetChannelMessage`	Allowed with Restriction	Allowed only for sent messages. Not allowed for messages in processing by channel flow unless you are the message sender.
`ListChannelMessages`	Allowed with Restriction	
`DeleteChannelMessage`	Denied	Denied
`RedactChannelMessage`	Denied	
`UpdateChannelMessage`	Denied	
`AssociateChannelFlow`	Denied	
`DisassociateChannelFlow`	Denied	
`GetChannelMessageStatus`	Allowed with Restriction	You can only get message status for your own messages.

Streaming messaging data

You can configure an `AppInstance` to receive data, such as messages and channel events, in the form of a stream. You can then react to that data in real time. Currently, Amazon Chime SDK messaging only accepts Kinesis streams as stream destinations. You must have these prerequisites to use Kinesis streams with this feature:

- Kinesis streams must be in the same AWS account as the `AppInstance`.
- A stream must be in the same region as the `AppInstance`.
- Stream names have a prefix that starts with `chime-messaging-`.
- You must configure at least two shards. Each shard can receive data up to 1MB per second, so scale your stream accordingly.
- You must enable server-side encryption (SSE).

To configure a Kinesis stream

1. Create one or more Kinesis streams using the prerequisites in the previous section, then get the ARN. Ensure the caller has Kinesis permissions in addition to Amazon Chime permissions.

   The following examples show how to use the AWS CLI to create a Kinesis stream with two shards, and how to enable SSE.

   ```
   aws kinesis create-stream --stream-name chime-messaging-unique-name --
   shard-count 2
   ```

   ```
   aws kinesis start-stream-encryption --stream-name chime-messaging-
   unique-name --encryption-type KMS --key-id "alias/aws/kinesis"
   ```

2. Configure streaming by calling the PutMessagingStreamingConfigurations API.

   You can configure one or both of two data types, and you can choose the same stream or separate streams for them.

   The following examples show how to use the AWS CLI to configure an `appinstance` to stream the `ChannelMessage` and `Channel` data types.

   ```
   aws chime-sdk-messaging put-messaging-streaming-configurations --app-instance-
   arn app_instance_arn \
   ```

```
--streaming-configurations
 DataType=ChannelMessage,ResourceArn=kinesis_data_stream_arn
```

```
aws chime-sdk-messaging put-messaging-streaming-configurations --app-instance-
arn app_instance_arn \
--streaming-configurations DataType=Channel,ResourceArn=kinesis_data_stream_arn
```

The data types have the following scopes:

DataType	Event types generated
ChannelMessage	CREATE_CHANNEL_MESSAGE
	REDACT_CHANNEL_MESSAGE
	UPDATE_CHANNEL_MESSAGE
	DELETE_CHANNEL_MESSAGE
Channel	CREATE_CHANNEL
	CREATE_SUB_CHANNEL
	UPDATE_CHANNEL
	DELETE_CHANNEL
	UPDATE_CHANNEL_EXPIRATION_SETTINGS
	DELETE_SUB_CHANNEL
	CREATE_CHANNEL_MEMBERSHIP

DataType	Event types generated	
	DELETE_CHANNEL_MEMBERSHIP	
	CREATE_CHANNEL_BAN	
	DELETE_CHANNEL_BAN	
	CREATE_CHANNEL_MODERATOR	
	DELETE_CHANNEL_MODERATOR	

3. Start reading the data from your configured Kinesis stream.

> **Note**
>
> Any events sent before you configure streaming are not sent to your Kinesis stream.

Data format

Kinesis outputs records in JSON format with the following fields: `EventType` and `Payload`. The payload format depends on the `EventType`. The following table lists the event types and their corresponding payload formats.

EventType	Payload format	
CREATE_CHANNEL_MESSAGE	Channel message	
REDACT_CHANNEL_MESSAGE		
UPDATE_CHANNEL_MESSAGE		

EventType	Payload format	
DELETE_CHANNEL_MESSAGE		
CREATE_CHANNEL	Channel	
UPDATE_CHANNEL		
DELETE_CHANNEL		
UPDATE_CHANNEL_EXPIRATION_SETTINGS		
CREATE_CHANNEL_MEMBERSHIP	ChannelMembership	
DELETE_CHANNEL_MEMBERSHIP		
CREATE_CHANNEL_BAN	ChannelBan	
DELETE_CHANNEL_BAN		
CREATE_CHANNEL_MODERATOR	ChannelModerator	
DELETE_CHANNEL_MODERATOR		
CREATE_SUB_CHANNEL	channelARN	
DELETE_SUB_CHANNEL	SubChannelId	

Using elastic channels to host live events

Elastic channels support large-scale chat experiences with up to 1-million members. Typical uses include watch parties for sporting or political events. You can use elastic channels only in the US East (N. Virginia) region.

An elastic channel consists of a single channel with a common configuration, plus a varying—or *elastic*—number of sub-channels. The configuration also includes minimum and maximum thresholds for the members in the sub-channels.

For example, say you create an elastic channel with 100 sub-channels, and for the sub-channels you set a low threshold of 500 members and a high threshold of 10,000 members. When users join this example channel, the system automatically assigns them to a single sub-channel until the member count exceeds 10,000. At that point, the system creates a new sub-channel and adds any new members there. As users leave, the system deletes sub-channels and distributes members across the remaining sub-channels.

Splitting the audience across sub-channels makes conversations easier for participants to follow. Moderators also have reduced workloads, because they only need to watch some of the sub-channels. In addition, moderators can use the built-in tools that elastic channels provide. For example, moderators can ban users from a channel, create moderators, and use channel flows to automatically moderate all the messages in the channel.

For more information about Amazon Chime SDK Messaging quotas, refer to Messaging Quotas in the *Amazon Chime SDK General Reference*.

Topics

Prerequisites

You must have the following to use elastic channels.

- Knowledge of Amazon Chime SDK Messaging functionality, such as managing channels, and sending and receiving messages.
- The ability to invoke the Amazon Chime SDK Messaging APIs.

Elastic channel concepts

To use elastic channels effectively, you must understand these concepts.

Sub-channels

Elastic channels divide their members into logical containers called sub-channels. When you add an `AppInstanceUser` to an elastic channel, the user becomes a member of a sub-channel. That user can send and receive messages, but only with other members of that sub-channel. The system never allows messages from one sub-channel to appear in other sub-channels.

Scaling

To support user engagement, every sub-channel must meet a minimum membership requirement. You provide that value when you create an elastic channel. As users join or leave an event, the system transfers members to different sub-channels, which makes the overall channel "elastic." Sub-channels run the following scaling actions.

- **SCALE_OUT** – When a new elastic channel membership request comes in and all sub-channels are full, the system scales out by creating a new sub-channel, and then transferring memberships from existing sub-channels to the new sub-channel.
- **SCALE_IN** – When a sub-channel membership count goes below the minimum requirement, and another sub-channel has the capacity to accommodate all the members of the first sub-channel, a `SCALE_IN` event transfers those memberships, and then deletes the sub-channel and all messages.

> **Note**
>
> If you need to access messages from channels that have been deleted, you must first turn on message streaming. For more information, refer to Streaming messaging data.

Member transfer

This occurs when membership balancing moves an `AppInstanceUser` from one sub-channel to another. The `AppInstanceUser` still belongs to the elastic channel after the transfer. However, the new sub-channel contains different memberships and messages, so the messages sent by the `AppInstanceUser` after the transfer go to those different members. Membership balancing does not affect moderator memberships.

> **Note**
>
> Elastic channels don't support hidden memberships, membership preferences, and read message timestamps.

Additional supported features

Elastic channels also support these messaging features.

- Prefetch
- Channel flows

Creating elastic channels

You use the `ElasticChannelConfiguration` field in the CreateChannel API to create an elastic channel. Once you create an elastic channel, you create channel memberships.

> **Note**
>
> - For non-elastic channels, the `AppInstanceUser` that creates the channel is automatically added to that channel as a member and moderator. For elastic channels, the channel creator is only added as a moderator.
> - You can't update an `ElasticChannelConfiguration` once set.
> - You can't update a channel from elastic to non-elastic and vice-versa.
> - You can't include a list of member ARNs in a CreateChannel API request. However, you can include a list of moderator ARNs.

- You can't create an `UNRESTRICTED` type elastic channel.

Managing elastic channel members

To manage the members in an elastic channel, use the CreateChannelMembership, CreateChannelModerator, and CreateChannelBan APIs. The following information explains how to use them.

Channel memberships

The `CreateChannelMembership` API creates memberships at the sub-channel level. sub-channels can include moderators and regular members.

- **Moderators** – You can add moderators to multiple sub-channels. That allows the moderators to send messages on each of the sub-channels they belong to. When you add a moderator to a sub-channel, you must provide the `SubChannelId`.

  If you want to assign moderators to new sub-channels automatically, you can enable message streaming, listen for sub-channel creation events, and then create a moderator membership in response to those events.

  Finally, you can delete moderators from specific sub-channels, or from all sub-channels. You use the DeleteChannelMembership API in both cases. To delete a moderator from a specific sub-channel, you provide the `SubChannelId`. If you don't provide an ID for a sub-channel, the system removes the moderator from all sub-channels. Finally, you can use the ListSubChannels API to list the sub-channels and the number of members in each.
- **Regular members** – These comprise the majority of channel memberships. You can only add a regular member to one sub-channel. Also, you can't pass a `SubChannelId` when creating or deleting channel memberships, because the system controls which sub-channel a membership is created in.

Channel moderators

The `CreateChannelModerator` API creates moderators at the elastic channel level. Moderators can view all messages in all sub-channels. When you promote a regular member to channel moderator, the system removes all existing channel memberships for that member. The same happens when you demote a moderator.

Channel bans

The `CreateChannelBan` API creates bans at the elastic channel level. A banned `AppInstanceUser` can't belong to any sub-channel. When you ban a member, the system removes all the channel memberships for that member.

Sending elastic channel messages

The SendChannelMessage API creates messages at the sub-channel level. To send messages, you must have a `subChannelId`. You can also use the UpdateChannelMessage, and RedactChannelMessage APIs to edit and delete messages, but in all cases, you must have a `subChannelId`.

> **Note**
>
> Message senders can only edit or redact messages if they belong the sub-channel that they send messages to. If membership balancing transfers a member to another sub-channel, that member can only edit or redact the messages that they send in that new sub-channel.

Understanding WebSocket system messages in elastic channels

The Amazon Chime SDK sends system messages to all connected clients for events that take place in channels. The following list describes the system messages for elastic channels.

Message events

Event payloads for elastic channels contain the `subChannelId` field. Payloads for non-elastic channels remain the same.

Membership events

The `CREATE_CHANNEL_MEMBERSHIP` and `DELETE_CHANNEL_MEMBERSHIP` events now have the `subChannelId` field in their payloads.

Elastic channels don't support the `BATCH_CREATE_CHANNEL_MEMBERHSIP` event. When you call the BatchCreateChannelMembership API, the system sends individual `CREATE_CHANNEL_MEMBERSHIP` events.

You can now use the `UPDATE_CHANNEL_MEMBERSHIP` event type to signal changes in membership information. For example, during a member transfer from one sub-channel to another, the system sends an `UPDATE_CHANNEL_MEMBERSHIP` event with the new `SubChannelId` in the payload to indicate that the member was transferred.

> **Note**
>
> The system only sends the `UPDATE_CHANNEL_MEMBERSHIP` event to the member that was transferred, and not to other members of the sub-channel. For this reason, we encourage you to use the ListChannelMemberships API instead of WebSockets to populate your channel membership rosters. For more information, refer to Using WebSockets to receive messages.

Using Kinesis streams to receive system messages

You can configure an `AppInstance` to receive data in the form of a stream. For example, a stream can include messages, sub-channel events, and channel events.

As part of that, we support the `CREATE_SUB_CHANNEL` and `DELETE_SUB_CHANNEL` events. They indicate when a sub-channel was created or deleted as part of membership balancing. For more information about receiving data streams, refer to Streaming messaging data.

Testing elastic channels in our demo app

You can test all of Amazon Chime SDK Messaging features on GitHub at https://github.com/aws-samples/amazon-chime-sdk/tree/main/apps/chat.

Using mobile push notifications to receive messages

You can configure Amazon Chime SDK Messaging to send channel messages to mobile push notification channels. The Amazon Chime SDK requires an Amazon Pinpoint application configured for push notifications. Your Amazon Pinpoint application must meet these prerequisites:

- Your Amazon Pinpoint application must have at least an FCM or APNS channel configured and enabled.
- Your Amazon Pinpoint application must reside in the same AWS account and region as your Amazon Chime SDK app instance.

> **Note**
>
> By default, all the members of a push notification channel receive the push notifications, including message senders. However, you can set a filter rule that prevents messages from going to senders. For more information, see Using filter rules to filter messages, later in this section.

Topics

Create an Amazon Pinpoint application

To send push notifications, the Amazon Chime SDK requires an Amazon Pinpoint application configured to send pushes to your mobile app. The following steps explain how to use the AWS console to create a Pinpoint application.

To create a Amazon Pinpoint application

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

   If this is your first time using Amazon Pinpoint, you see a page that introduces you to the features of the service.
2. In the **Get started** section, enter a name for your project, and then choose **Create a project**.
3. On the **Configure features** page, next to **Push Notifications** choose **Configure**.
4. On the **Set up push notifications** page, toggle **Apple Push Notification service (APNs)**, **Firebase Cloud Messaging (FCM)**, or both, and complete the required fields.

> **Important**
>
> The Amazon Chime SDK currently only supports sending push notifications to APNs and FCM.

5. When finished, choose **Save.**
6. Return to the Amazon Pinpoint console at https://console.aws.amazon.com/pinpoint/ and note the **Project ID** value. You use that as the ARN for your Amazon Pinpoint application.

Create a service role

AWS uses service roles to grant permissions to an AWS service so it can access AWS resources. The policies that you attach to a service role determine which resources the service can access and what it can do with those resources. The service role that you create for the Amazon Chime SDK gives the service permission to make `SendMessages` calls to your Amazon Pinpoint application.

To create a service role

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 **Policies**, and then choose **Create Policy**.
3. Choose the **JSON** tab and copy the policy below into the text box. Be sure to replace `project_id` with the the ID of the Amazon Pinpoint application created in the previous step, and the `aws_account_id` with your AWS Account ID.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Action": "mobiletargeting:SendMessages",
        "Resource": "arn:aws:mobiletargeting:region:aws_account_id:apps/project_id/
messages",
        "Effect": "Allow"
    }
}
```

4. Choose **Next: Tags**.

5. Choose **Next: Review**, and enter **`AmazonChimePushNotificationPolicy`** in the **Name** field, and choose **Create Policy**.
6. In the navigation pane, choose **Roles**, and then choose **Create role**.
7. On the **Create role** page, choose **AWS service**, open the **Choose a user case** list and choose **EC2**.
8. Choose **Next: Permissions**, and in the search box, enter **`AmazonChimePushNotificationPolicy`**, and select the check box next to the policy.
9. Choose **Next: Tags**.
10. Choose **Next: Review**, and enter **`ServiceRoleForAmazonChimePushNotification`** in the **Name** field.

> **Important**
>
> You must use the name listed above. The Amazon Chime SDK only accepts that specific name.

11. Choose **Create role**, and on the **Roles** page. enter **`ServiceRoleForAmazonChimePushNotification`** in the search box, and choose the matching role.
12. Choose the **Trust Relationships** tab, choose **Edit trust relationship** and replace the existing policy with the one below.

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "messaging.chime.amazonaws.com"
             },
             "Action": "sts:AssumeRole"
         }
    ]
}
```

13. Choose **Update Trust Policy**.

> **Important**
>
> Modifying the role by changing the name, the permission policy, or the trust policy can break the push notification functionality.

Register a mobile device endpoint as an App Instance user

To receive push notifications, app instance users must first use the RegisterAppInstanceUserEndpoint API to register a mobile device. They must register from a mobile app that has access to the device token for the device's operating system.

To ensure the app instance User has access to the Amazon Pinpoint application listed in the ARN, the user must have permission to call `mobiletargeting:GetApp` on the Amazon Pinpoint ARN. Otherwise, the Amazon Chime SDK throws a 403 Forbidden error when calling RegisterAppInstanceUserEndpoint.

This example shows the policy needed to register an endpoint.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionToRegisterEndpoint",
            "Effect": "Allow",
            "Action": "chime:RegisterAppInstanceUserEndpoint",
            "Resource": "arn:aws:chime:region:aws_account_id:app-
instance/app_instance_id/user/app_instance_user_id"
        },
        {
            "Sid": "PermissionToGetAppOnPinpoint",
            "Effect": "Allow",
            "Action": "mobiletargeting:GetApp",
            "Resource": "arn:aws:mobiletargeting:region:aws_account_id:apps/project_id"
        }
    ]
}
```

To register an endpoint

- Use the Amazon Pinpoint ARN and your device token to call the RegisterAppInstanceUserEndpoint API.

Send a channel message with notifications enabled

The SendChannelMessage API has an optional `PushNotification` attribute that the Amazon Chime SDK uses to build the push notification to send to Amazon Pinpoint. Currently, the Amazon Chime SDK only supports the notification title and body fields.

The Amazon Chime SDK also supports APNs VoIP pushes. To send a push notification as an APNs VoIP push, set the type in the `PushNotification` attribute to VOIP.

Receiving push notifications

Along with the channel message push notification title and body, the Amazon Chime SDK also includes the channel message ID and channel ARN in the data payload. You use that information to load the full channel message.

The following examples shows a typical push notification payload.

```
{
    "pinpoint.openApp=true",
    "pinpoint.notification.title=PushNotificationTitle",
    "pinpoint.notification.body=PushNotificationBody",
    "pinpoint.campaign.campaign_id=_DIRECT",
    "pinpoint.notification.silentPush=0",
    "pinpoint.jsonBody="{
        "chime.message_id":"ChannelMessageId",
        "chime.channel_arn":"ChannelARN"
    }
}
```

Disabling or filtering push notification receipts

The Amazon Chime SDK provides multiple options to allow app instance users to control whether they wish to receive push notifications.

Disabling all push notifications

App instance users can disable push notifications entirely by calling UpdateAppInstanceUserEndpoint and setting the `AllowMessages` attribute to `NONE`.

Disabling push notifications for a channel

App instance users can disable push notifications for a specific channel by calling PutChannelMembershipPreferences to `NONE` in the **PushNotification Preferences** field.

Filtering push notifications for a channel

App Instance users can set a filter rule so they only receive specific push notifications using the PutChannelMembershipPreferences API. For more information, refer to Using filter rules to filter messages.

Debugging push notification failures

The Amazon Chime SDK integrates with Amazon EventBridge in order to notify you of push message delivery failures. To further debug failures, you can also look into the CloudWatch metrics that Amazon Pinpoint sends for failures.

The following table lists and describes the delivery error messages.

Message	Description
The request processing has failed because of an unknown error, exception or failure.	We encountered an internal error. Please try again.
The specified resource was not found. AppInstanceUserEndpoint will be deactivated.	The Amazon Pinpoint application does not exist.
Too many requests sent to Amazon Pinpoint.	Amazon Pinpoint has throttled your outgoing messages.
Unable to send messages. Please verify IAM Permissions Policy on ServiceRoleForAmazonChimePushNotification.	The role created for the Amazon Chime SDK does not have permission to call `mobiletargeting:SendMessages`. Please verify the IAM policy on the role.
Unable to send messages. Please verify IAM Trust Relationships on ServiceRoleForAmazonChimePushNotification.	The Amazon Chime SDK does not have permission to access the role for push notifications.

Message	Description
	Please verify the IAM role's trust policy contains the service principal, `messaging.chime.amazonaws.com`.

Using filter rules to filter messages

The Amazon Chime SDK supports setting filter rules on an app instance user's channel membership to limit which message they will receive. Filter rules are set on the channel membership and run against the message attributes map. The message attribute map must be a map of string keys to string values. Filter rules support inclusion and exclusion with exact string matching.

> **Important**
>
> - The Amazon Chime SDK only supports escaped JSON strings as the filter rule.
> - All members of a notification channel receive the push notifications, including message senders. To prevent that from happening, see the first example rule below.

To set filter rules on a channel membership, use the PutChannelMembershipPreferences API. You can include message attributes in a channel message as part of the SendChannelMessage API call.

Topics

Filter rule types

The Amazon Chime SDK supports the following types of filter rules:

- Inclusive exact string matching
- Exclusive exact string matching
- Multiple filter rules using AND or OR

Filter rule limits

The Amazon Chime SDK imposes the following restrictions on filter rules:

- We only support exact string matching.
- A total filter rules size of 2KB.
- A total message attribute size of 1KB.
- A maximum of five (5) separate constraints inside an OR filter rule.
- A maximum complexity of 20 for the entire filter rule. *Complexity* is calculated as the sum of the number of keys and values in the filter rule:

  For example, this filter rule has a complexity of 4.

  ```
  "FilterRule": "{\"type\":[{\"anything-but\": [\"Room\"]}],\"mention\":[\"Bob\"]}
  ```

  We calculate that value as follows:

  ```
  Keys = “type” and “mention” - Complexity 2
  Values = "Room" and "Bob" -   Complexity 2

                          Total complexity = 4
  ```

Example filter rules

The following examples show several ways to use channel membership preferences and filter rules.

Preventing messages from going to senders

This filter rule sends messages to all channel members except the message sender.

```
{
    "Preferences": {
        "PushNotifications": {
            "FilterRule": "{\"type\":[{\"anything-but\": [\"USER_ARN\"]}]}",
            "AllowNotifications": "FILTERED"
        }
    }
}
```

App instance users with the preferences shown above receive a channel message with the following attributes:

```
"MessageAttributes": {
    "senderId": {
        "StringValues": ["USER_ARN"]
    }
}
```

Inclusive string matching

This filter rule allows any message with the message attribute key “mention” and the value “Bob.”

```
{
    "Preferences": {
        "PushNotifications": {
            "FilterRule": "{\"mention\":[\"Bob\"]}",
            "AllowNotifications": "FILTERED"
        }
    }
}
```

An app instance user with the preferences shown above receives a channel message with the following message attributes:

```
"MessageAttributes": {
    "mention": {
        "StringValues": ["Bob", "Alice"]
    }
}
```

However, the app instance user will not receive a channel message with the following attributes:

```
"MessageAttributes": {
    "mention": {
        "StringValues": ["Tom"]
    }
}
```

Exclusive string matching

This filter rule allows any message except those containing the attribute key “type” and the value “Room”.

```
{
    "Preferences": {
        "PushNotifications": {
            "FilterRule": "{\"type\":[{\"anything-but\": [\"Room\"]}]}",
            "AllowNotifications": "FILTERED"
        }
    }
}
```

An app instance user with those preferences receives a channel message with the following message attributes:

```
"MessageAttributes": {
    "type": {
        "StringValues": ["Conversation"]
    }
}
```

However, the app instance user doesn't see a channel message with the following attributes:

```
"MessageAttributes": {
    "type": {
        "StringValues": ["Room"]
    }
}
```

A multiple filter rule with AND logic

When you combine filter rules with AND logic, a message must meet all the filter criteria for the filter to apply.

```
{
    "Preferences": {
        "PushNotifications": {
            "FilterRule": "{\"type\":[{\"anything-but\": [\"Room\"]}],\"mention\":
[\"Bob\"]}",
            "AllowNotifications": "FILTERED"
        }
```

```
    }
}
```

An app instance user with the preferences above receives a channel message with the following message attributes:

```
"MessageAttributes": {
    "mention": {
        "StringValues": ["Bob"]
    },
    "type": {
        "StringValues": ["Conversation"]
    }
}
```

A multiple filter rule with OR logic

You use `$or` to combine filter rules with OR logic. When you use OR logic, a message must meet one of the criteria for the filter to apply.

```
{
    "Preferences": {
        "PushNotifications": {
            "FilterRule": "{\"$or\":[{\"mention\":[\"Bob\"]},{\"type\":[{\"anything-but
\": [\"Room\"]}]}]}",
            "AllowNotifications": "FILTERED"
        }
    }
}
```

An app instance user with the preferences above receives a channel message with the following message attributes:

```
"MessageAttributes": {
    "mention": {
        "StringValues": ["Bob"]
    }
}
```

An app instance user with the preferences above receives a channel message with the following message attributes:

```
"MessageAttributes": {
    "type": {
        "StringValues": ["Conversation"]
    }
}
```

Using service-linked roles

The Amazon Chime SDK uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that links directly to Amazon Chime SDK. Amazon Chime SDK predefines the service-linked roles, and they include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Amazon Chime SDK more efficient, because you aren't required to manually add the necessary permissions. Amazon Chime SDK defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon Chime SDK can assume its roles. The defined permissions include the trust and permissions policies. The 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 Chime SDK 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. Look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the documentation for that service.

Topics

- Using service-linked roles for data streaming

Using service-linked roles for data streaming

The following sections explain how to manage the service-linked role for data streaming.

Topics in this section

- Service-linked role permissions
- Creating a service-linked role

- Editing a service-linked role
- Deleting the resources used by a service-linked role
- Deleting a service-linked role

Service-linked role permissions

Amazon Chime SDK uses the service-linked role named **AWSServiceRoleForChimeSDKMessaging**. The role grants access to the AWS services and resources used or managed by Amazon Chime SDK, such as the Kinesis streams used for data streaming.

The **AWSServiceRoleForChimeSDKMessaging** service-linked role trusts the following services so that those services can assume the role:

- messaging.chime.amazonaws.com

The role permissions policy allows Amazon Chime SDK to complete the following actions on the specified resource:

- `kms:GenerateDataKey` only when the request is made using `kinesis.*.amazonaws.com`.
- `kinesis:PutRecord`, `kinesis:PutRecords`, or `kinesis:DescribeStream` only on streams of the following format: `arn:aws:kinesis:*:*:stream/chime-messaging-*`.

The following example shows the policy.

```
{
     "Version": "2012-10-17",
     "Statement": [
      {
       "Effect": "Allow",
       "Action": [
        "kms:GenerateDataKey"
       ],
       "Resource": "*",
       "Condition": {
        "StringLike": {
         "kms:ViaService": [
          "kinesis.*.amazonaws.com"
         ]
        }
```

```
        }
      },
      {
       "Effect": "Allow",
       "Action": [
        "kinesis:PutRecord",
        "kinesis:PutRecords",
        "kinesis:DescribeStream"
       ],
       "Resource": [
        "arn:aws:kinesis:*:*:stream/chime-messaging-*"
       ]
      }
     ]
    }
```

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

You don't need to manually create a service-linked role. When you use the PutMessagingStreamingConfigurations API to create a data streaming configuration, Amazon Chime SDK creates the service-linked role for you.

You can also use the IAM console to create a service-linked role with the Amazon Chime SDK use case. In the AWS CLI or the AWS API, create a service-linked role with the `messaging.chime.amazonaws.com` service name. For more information, see Creating a service-linked role in the *IAM user Guide*. If you delete this role, you can repeat this process to create it again.

Editing a service-linked role

After you create a service-linked role, you can only edit its description, and you do that using IAM. For more information, see Editing a service-linked role in the *IAM user Guide*.

Deleting the resources used by a service-linked role

Before you can use IAM to delete a service-linked role, you must first delete any resources used by the role.

> **Note**
>
> Deletions can fail if you try to delete resources while Amazon Chime SDK is using them. If a deletion fails, wait a few minutes and try the operation again.

To delete resources used by the AmazonChimeServiceChatStreamingAccess role

Run the following CLI command to turn off data streaming for the app instance:

- `aws chime-sdk-messaging delete-messaging-streaming-configurations --app-instance-arn` *`app_instance_arn`*

This action deletes all streaming configurations for your app instance.

Deleting a service-linked role

When you no longer need a feature or service that requires a service-linked role, it's a best practice to delete that role. Otherwise, you have an unused entity that is not actively monitored or maintained. However, you must delete the resources used by your service-linked role before you can manually delete the role.

You can use the IAM console, AWS CLI, or the AWS API to delete the **AmazonChimeServiceRoleForChimeSDKMessaging** service-linked role. For more information, see Deleting a service-linked role in the IAM user Guide.

Using channel flows to process messages

You use channel flows to run business logic on in-flight messages before they're delivered to recipients in a messaging channel. Channel flows can perform actions such as removing government ID numbers, phone numbers, or profanity from messages. You can also use channel flows to perform functions such as aggregating responses to a poll before sending the results back to participants.

Prerequisites

- Knowledge of basic Amazon Chime SDK functionality, such as managing channels, and sending and receiving messages.
- The ability to invoke the Amazon Chime SDK messaging APIs.

Channel flow concepts

To use channel flows effectively, you must understand these concepts:

Channel processor

An AWS Lambda function that runs preprocessing logic on channel messages. When you associate a channel with a channel flow, the processor in the flow is invoked for every message in the channel. To reduce latency, a single processor works best for most use cases. Finally, each processor must make a callback to the Amazon Chime SDK service once processing completes.

> **Note**
>
> We currently only support one processor per channel flow. If you need more than one processor, submit a support ticket for an increase.

Channel flow

Channel Flows are containers for up to three channel processors, plus a run sequence. You associate a flow with a channel, and the processor takes action on all messages sent to that channel.

Invoking channel flows

The following items invoke channel flows:

- New persistent standard messages
- New non-persistent standard messages
- Updated persistent standard messages

> **Note**
>
> Channel flows don't process Control or System messages. For more information about the message types provided by Amazon Chime SDK Messaging, refer to Understanding message types.

Topics

Setting up a Channel Processor

To start using channel flows, you first create a processor Lambda function to handle preprocessing for your use case. For example, you can update message content or metadata, deny messages and prevent them from being sent, or let the original message through.

Prerequisites

- The Lambda function must be in the same AWS account and the same AWS Regions as the AppInstance.

Granting invocation permissions

You must give the Amazon Chime SDK messaging service permission to invoke your Lambda resource. For more information about permissions, refer to Using resource-based policies for AWS Lambda. For example:

Principal: "messaging.chime.amazonaws.com"

Action: lambda:InvokeFunction

Effect: Allow

AWS:SourceAccount:*Your AWSAccountId*.

AWS:SourceArn: "`arn:aws:chime:`*region*`:`*AWSAccountId*`:` *appInstance*/"

> **Note**
>
> You can provide a specific app instance ID to invoke your processor, or use a wildcard to allow all Amazon Chime SDK app instances in an account to invoke your processor.

Granting callback permissions

You also need to allow your processor Lambda functions to call the `ChannelFlowCallback` API. For information about doing that, see AWS Lambda run role in the *AWS Lambda developer guide*.

You can add an Inline policy to your Lambda function's run role. This example allows the processor to invoke the `ChannelFlowCallback API`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "chime:ChannelFlowCallback"
            ],
            "Resource": [
                "arn:aws:chime:Region:AwsAccountId:appInstance/*"
            ]
        }
    ]
}
```

> **Note**
>
> Follow the best practices for Lambda functions. For more information, refer to these topics:
>
> - Performance Efficiency Best Practices
> - Best practices for working with AWS Lambda
> - Configuring reserved concurrency
> - Asynchronous invocation

Invoking processor Lambda functions

When a user sends a message, the following input request invokes the processor Lambda function.

```
{
    "EventType": "string"
    "CallbackId": "string"
```

```
    "ChannelMessage": {
        "MessageId": "string",
        "ChannelArn": "string",
        "Content": "string",
        "Metadata": "string",
        "Sender":{
            "Arn": "string",
            "Name": "string"
        },
        "Persistence": "string",
        "LastEditedTimestamp": "string",
        "Type": "string",
        "CreatedTimestamp": "string",
    }
}
```

EventType

The event being sent to processor. The value is a `CHANNEL_MESSAGE_EVENT` constant.

CallbackId

The token used while calling the `ChannelFlowCallback` API from the processor.

ChannelMessage

ChannelArn The ARN of the channel

Content Message content to be processed

CreatedTimestamp The time at which the message was created

LastEditedTimestamp The time at which a message was edited

MessageId The message identifier

Metadata Message metadata to be processed

Persistence Boolean that controls whether the message is persisted on the back end. Valid Values: `PERSISTENT | NON_PERSISTENT`

Sender The message sender. Type: an identity object.

Type The message type. ChannelFlow only supports the `STANDARD` message types. Valid Value: `STANDARD`

The processor function determines the following about each message.

- Whether to update the message content, metadata, or both
- Whether to deny a message
- Whether to leave a message unchanged

When processing finishes, the processor Lambda function sends the result back to the Amazon Chime SDK Messaging service so the message can be sent to all recipients. Message status is marked `PENDING` until the processor Lambda function sends back the results. The processor Lambda function has 48 hours to send back the results. We do not guarantee message delivery after that, and the ChannelFlowCallback API throws a Forbidden Exception error message. To send back the results, invoke the `ChannelFlowCallback` API.

Creating a channel flow

Once you have processor(s) setup, you use the Amazon Chime SDK Messaging APIs to create a channel flow. You can use a `Fallback` action to define whether to stop or continue processing if the channel flow can't connect to the processor Lambda function. If a processor has a fallback action of `ABORT`, the processor sets the message status to `FAILED`, and it doesn't send the message. Note that if the last processor in the channel flow sequence has a fallback action of `CONTINUE`, the message is considered processed and sent to recipients in the channel. Once you create a channel flow, you can associate it with individual channels. For more information, refer to the CreateChannelFlow API documentation.

Associating and disassociating channel flows

When you associate a channel is associated with a channel flow, the processor(s) in the channel flow pre-process all messages sent to the channel. You must be a channel moderator or administrator to invoke the channel flow association and disassociation APIs. Remember these facts as you go.

- You can associate a maximum of 1 channel flow with a channel at any given time. To associate a channel flow, call the AssociateChannelFlow API.
- To disassociate a channel flow and stop preprocessing of channel messages, call the DisassociateChannelFlow API.

Sending messages

You use the `SendChannelMessage` API to send messages to a channel. For a channel associated with a channel flow, the processor assigns one of the following status values.

Message status	Description
`SENT`	Message processed successfully.
`PENDING`	Ongoing processing.
`FAILED`	Processing failed because the processor Lambda function is unreachable.
`DENIED`	The message won't be sent.

Receiving intermediate status events

Websocket events

Websocket events are sent to a channel after they successfully establish a connection. For more information, refer to Using WebSockets to receive messages.

Event type	Status	Recipients	Notes
`CREATE_CHANNEL_MESSAGE`	`SENT`	All channel members	`SendChannelMessage` API with successful preprocessing
`UPDATE_CHANNEL_MESSAGE`	`SENT`	All channel members	`UpdateChannelMessage` API with successful preprocessing
`PENDING_CREATE_CHANNEL_MESSAGE`	`PENDING`	Message sender only	`SendChannelMessage`

Event type	Status	Recipients	Notes
			API with ongoing preprocessing
PENDING_UPDATE_CHANNEL_MESSAGE	PENDING	Message sender only	UpdateChannelMessage API with ongoing preprocessing
FAILED_CREATE_CHANNEL_MESSAGE	FAILED	Message sender only	SendChannelMessage API with failed preprocessing
FAILED_UPDATE_CHANNEL_MESSAGE	FAILED	Message sender only	UpdateChannelMessage API with failed preprocessing
DENIED_CREATE_CHANNEL_MESSAGE	DENIED	Message sender only	SendChannelMessage API with processor denying the message
DENIED_UPDATE_CHANNEL_MESSAGE	DENIED	Message sender only	UpdateChannelMessage API with processor denying the message

GetChannelMessageStatus API

This API provides an alternative way to retrieve message status if the event was not received due to a bad websocket connection. For more information, refer to the GetChannelMessageStatus API documentation.

> **Note**
>
> This API does not return statuses for denied messages, because we don't store them.

Creating failure alerts by automating with EventBridge

The Amazon Chime SDK delivers Events when there is an error invoking your processor Lambda function. Events are sent regardless of the `Fallback` action specified for the processor when creating a channel flow. You can write simple rules to specify these events, plus the automated actions to take when any of those events matches a rule. For more information, see the Amazon EventBridge User Guide. When errors like these occur, then depending on the `Fallback` action that you configure, members on the channel can't send messages, or messages will flow through the channel with no processing. For more information about the `Fallback` action, refer to Processor in the Amazon Chime SDK API reference.

This example shows a typical failure event.

```
{
    "version": "0",
    "id": "12345678-1234-1234-1234-111122223333",
    "detail-type": "Chime ChannelFlow Processing Status",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "region",
    "resources": [],
    "detail": {
        "eventType": "ProcessorInvocationFailure",
        "appInstanceArn": "arn:aws:chime:region:AWSAccountId:app-
instance/AppInstanceId",
        "channelArn": "arn:aws:chime:region:AWSAccountId:app-instance/AppInstanceId/
channel/ChannelId",
        "messageId":
 "298efac7298efac7298efac7298efac7298efac7298efac7298efac7298efac7",
        "processorResourceArn":
 "arn:aws:lambda:region:AWSAccountId:function:ChannelFlowLambda",
        "failureReason": "User is not authorized to perform: lambda:InvokeFunction on
 resource: arn:aws:lambda:region:AppInstanceId:function:ChannelFlowLambda because no
 resource-based policy allows the lambda:InvokeFunction action"
      }
```

```
}
```

Using AppInstanceBots as intelligent channel agents

You can use `AppInstanceBots` as intelligent channel agents. The agents recognize key phrases sent via `ChannelMessages` by channel members. The bot's natural language understanding model resolves the messages. In turn, that allows one or more channel members to engage in a natural language dialog defined by the bot's model. You supply the bots, so you control the depth of dialog and integration with your enterprise systems.

Prerequisites

- Knowledge of basic Amazon Chime SDK functionality, such as creating `AppInstanceUsers`, managing channels, and sending and receiving messages.
- The ability to invoke the Amazon Chime SDK Messaging APIs.
- Knowledge of basic Amazon Lex V2 functionality, such as creating an Amazon Lex V2 Bot, modeling intents and slots, creating bot versions, aliases, using session state, and Lambda hook integration.

> **Important**
>
> Use of Amazon Lex V2 is subject to the AWS Service Terms, including the terms specific to the AWS Machine Learning and Artificial Intelligence Services.

Topics

Creating an Amazon Lex V2 bot

To use AppInstance bots as agents, you first need to create an Amazon Lex V2 bot to manage the dialog interaction for an intelligent-agent scenario. To get started building an Amazon Lex V2 bot, see Getting Started with Amazon Lex V2 in the *Amazon Lex V2 Developer Guide*. For information about migrating an Amazon Lex V1 bot to Amazon Lex V2, see the Amazon Lex V1 to V2 migration guide.

Topics

Prerequisites

Your Amazon Lex V2 bot must have the following prerequisites.

- You must create the bot in an AWS Region that supports Amazon Lex V2 runtime endpoints.
- You must create the bot in the same AWS account and Region as the `AppInstance` and `AppInstanceBot`.
- The bot must grant invocation permissions via a resource-based policy to the `messaging.chime.amazonaws.com` service principal.
- The bot can model a Welcome Intent. This allows `AppInstanceBot` to announce itself and its capabilities upon membership in a channel.
- The bot should have a production version and aliases in order to configure the `AppInstanceBot`.
- The bot must use a supported language and locale. For more information about languages and locales, see Languages and locales supported in Amazon Lex V2 in the *Amazon Lex V2 Developer Guide*.

Granting invocation permissions

For an `AppInstanceBot` to invoke an Amazon Lex V2 Bot, the Amazon Chime SDK messaging service principal must have permission to invoke the Amazon Lex Bot resource. For more information about Amazon Lex V2 resource-based policy permissions, see Resource-based policy examples for Amazon Lex V2 in the *Amazon Lex V2 Developer Guide*.

The following example shows a resource-based policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "messaging.chime.amazonaws.com"
      },
      "Action": [
        "lex:PutSession",
        "lex:DeleteSession",
        "lex:RecognizeText"
      ],
      "Resource": "arn:aws:lex:region:aws-account-id:bot-alias/lex-bot-id/lex-bot-
alias-id",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "aws-account-id"
        },
        "ArnEquals": {
          "AWS:SourceArn": "arn:aws:chime:region:aws-account-id:app-instance/app-
instance-id/bot/app-instance-bot-id"
        }
      }
    }
  ]
}
```

> **Note**
>
> To allow one `AppInstanceBot` to invoke an Amazon Lex V2 bot, use the AppInstanceBot's ID. To allow all `AppInstanceBots` within an `AppInstance` to invoke an Amazon Lex V2 bot, use a wildcard. For example:

```
arn:aws:chime:region:aws-account-id:app-instance/app-instance-id/bot/
*
```

Creating a welcome intent

If you add an optional welcome intent to your Amazon Lex V2 bot model, your `AppInstanceBot` can announce itself and its capabilities when it joins a channel. The welcome intent can display a message, or it can initiate a dialog with channel members. The name of the of welcome intent can vary, and you define it in the AppInstanceBot's configuration.

For more information about intents, see Adding intents in the *Amazon Lex V2 Developer Guide*

Creating Amazon Lex V2 bot versions

When you create an Amazon Lex V2 Bot, you only create a *draft* version. The draft is a working copy of the bot that you can update. By default, the draft version is associated with an alias called `TestBotAlias`, and you should only use the draft bot for manual testing.

After you finish dialog modeling and building the draft bot, you create one or more *versions*, numbered snapshots of the draft Lex bot. Versions allow you to control the implementation that your client applications use. For example, you can publish versions for use in different parts of your workflow, such as development, beta deployment, and production.

For more information about Lex bot versioning, see Creating versions in the *Amazon Lex V2 Developer Guide*.

Creating Amazon Lex V2 bot aliases

Once you create one or more versions of an Amazon Lex V2 bot, you create *aliases*. Aliases act as named pointers to the versions of an Amazon Lex V2 bot. For example, You can only associate an alias with one version at a time.

For more information about Lex bot aliases, see Creating aliases in the *Lex V2 Developer Guide*.

Setting up AppInstance bots

After you have an Amazon Lex V2 bot with a model, version, and alias, you use the Amazon Chime SDK messaging APIs or the CLI to create an AppInstanceBot. For more information about using the APIs, see to the CreateAppInstanceBot API documentation.

> **Note**
>
> You use the `InvokedBy` attribute to configure the dialog interaction behavior of the `AppInstanceBot`. You can configure the types of message that trigger a bot, such as standard messages or targeted messages.

The following example shows how to use the AWS CLI to create an AppInstanceBot that all standard messages with `MENTIONS`, and targeted messages, can invoke.

```
aws chime-sdk-identity create-app-instance-bot \
--app-instance-arn app-instance-arn \
--name app-instance-bot-name \
--configuration '{
   "Lex": {
      "LexBotAliasArn": "lex-bot-alias-arn",
      "LocaleId": "lex_bot_alias_locale_id",
      "InvokedBy": {
          "StandardMessages": "MENTIONS",
          "TargetedMessages": "ALL"
      }
      "WelcomeIntent": "welcome-intent-name"
   }
}
```

Creating a channel membership for an AppInstanceBot

Once you create the AppInstanceBot, you add it as a member to a new or existing channel. For more information see CreateChannel and CreateChannelMembership in the *Amazon Chime SDK messaging API* documentation.

The following example shows how to use the AWS CLI to create a channel and add an `AppInstanceBot` as a member.

```
aws chime-sdk-messaging create-channel \
--chime-bearer caller_app_instance_user_arn \
--app-instance-arn app_instance_arn \
--name channel_name \
--member-arns '[
   "app_instance_bot_arn"
```

```
]'
```

The following example shows how to use the AWS CLI to add an `AppInstanceBot` to an existing channel.

```
aws chime-sdk-messaging create-channel-membership \
--chime-bearer caller_app_instance_user_arn \
--channel-arn channel_arn \
--member-arn app_instance_bot_arn
```

Sending messages to an AppInstanceBot

You use the SendChannelMessage API to send messages to an AppInstanceBot. You send the messages to the channel in which the AppInstanceBot is a member. If the natural language understanding model recognizes the message content and elicits an Amazon Lex intent, the AppInstanceBot responds with a channel message and initiates a dialog.

You can also send target messages to a member of the channel, which could be an AppInstanceUser or an AppInstanceBot. Only the target and the sender can view targeted messages. Only users who can see targeted messages can take actions on them. However, administrators can delete targeted messages that they can't see.

The following example shows how to use the AWS CLI to send a channel message.

```
aws chime-sdk-messaging send-channel-message \
--chime-bearer caller_app_instance_user_arn \
--channel-arn channel_arn \
--content content \
--type STANDARD \
--persistence PERSISTENT
```

Processing messages from Amazon Lex

When sending messages to Amazon Lex, Amazon Chime SDK Messaging populates the `CHIME.channel.arn` and `CHIME.sender.arn` with the channel and sender's ARN information as request attributes. You can use the attributes to determine who sent a message and the channel the sender belongs to. For more information, refer to Enabling custom logic with AWS Lambda functions in the *Amazon Lex Developer Guide*.

Processing responses from an AppInstanceBot

When a user sends a message, the AppInstanceBot responds with a channel message. You can list channel messages to get the bot's response.

The following example shows you to use the CLI to list channel messages.

```
aws chime-sdk-messaging list-channel-messages \
--chime-bearer caller_app_instance_user_arn \
--channel-arn channel_arn
```

Success responses from an AppInstanceBot take the following format.

```
{
    "MessageId": "messageId",
    "Content": "*{\"Messages\":[{\"...\"}]}*",
    "ContentType": "application/amz-chime-lex-msgs",
    "MessageAttributes": {
        "CHIME.LEX.sessionState.intent.name": {
            "StringValues": [
                "lex_bot_intent_name"
            ]
        },
        "CHIME.LEX.sessionState.intent.state": {
            "StringValues": [
                "lex_bot_intent_fullfilment_status"
            ]
        },
        "CHIME.LEX.sessionState.originatingRequestId": {
            "StringValues": [
                "lex_bot_originating_request_id"
            ]
        },
        "CHIME.LEX.sessionState.sessionId": {
            "StringValues": [
                "lex_bot_session_id"
            ]
        }
    },
    "Sender": {
        "Arn": "app_instance_bot_arn",
        "Name": "app_instance_bot_name"
    },
```

```
    "Type": "STANDARD",
}
```

Content

The `Content` field contains a list of messages originating from the Amazon Lex V2 bot. For more information about those messages, refer to messages in the Amazon Lex V2 `RecognizeText` API.

The following example shows how to use the `Content` field in a welcome message.

```
{
    "Messages":
    [
        {
            "Content": "Hello!",
            "ContentType": "PlainText"
        },
        {
            "ContentType": "ImageResponseCard",
            "ImageResponseCard":
            {
                "Title": "Hello! I'm BB, the Bank Bot.",
                "Subtitle": "I can help you with the following transactions",
                "Buttons":
                [
                    {
                        "Text": "Check balance",
                        "Value": "Check balance"
                    },
                    {
                        "Text": "Escalate to agent",
                        "Value": "Escalate to agent"
                    }
                ]
            }
        }
    ]
}
```

For a failure response, the Content field contains an error message and code in the following format:

```
{
    "Code": error_code
}
```

ContentType

The `ContentType` refers to the type of payload the `Content` field contains, and must be checked to parse the `Content` field.

> **Note**
>
> The Lex V2 bot uses a different `ContentType`.

`ContentType` is set to `application/amz-chime-lex-msgs` for a success response, or `application/amz-chime-lex-error` for a failure response.

MessageAttributes

A *MessageAttribute* is a map of string keys to string values. A response from an `AppInstanceBot` contains the following message attributes mapped to a response from an Amazon Lex bot.

- **CHIME.LEX.sessionState.intent.name** – The name of the Lex bot intent that the request attempted to fulfill.
- **CHIME.LEX.sessionState.intent.state** – The current state of the intent. Possible values include: `Fulfilled`, `InProgress`, and `Failed`.
- **CHIME.LEX.sessionState.originatingRequestId** – A unique identifier for a specific request to Amazon Lex bot. This is set to the `MessageId` of the originating user message that triggered the AppInstanceBot.
- **CHIME.LEX.sessionState.sessionId** – A unique identifier for a conversation between the user and the bot. When a user starts a chat with your bot, Amazon Lex creates a session.

For more information about Amazon Lex sessions and session states, refer to SessionState in the *Amazon Lex API Reference*, and Managing sessions in the *in the Amazon Lex V2 Developer Guide*

For more information about the attributes that Amazon Lex V2 returns, refer to the Amazon Lex Runtime V2 APIs.

Using rules to send events to Amazon EventBridge

The Amazon Chime SDK delivers EventBridge events when an error prevents it from invoking the Amazon Lex V2 Bot. You can create EventBridge rules that recognize those events and automatically take action when the rule is matched. For more information, see Amazon EventBridge rules in the *Amazon EventBridge User Guide*.

The following example shows a typical failure event.

```
{
  version: '0',
  id: '12345678-1234-1234-1234-111122223333',
  'detail-type': 'Chime Messaging AppInstanceBot Lex Failure',
  source: 'aws.chime',
  account: 'aws-account-id',
  time: 'yyyy-mm-ddThh:mm:ssZ',
  region: "region",
  resources: [],
  detail: {
    resourceArn: 'arn:aws:chime:region:aws-account-id:app-instance/app-instance-id/
bot/app-instance-bot-id',
    failureReason: "1 validation error detected: Value at 'text' failed to satisfy
 constraint: Member must have length less than or equal to 1024 (Service: LexRuntimeV2,
 Status Code: 400, Request ID: request-id)"
  }
}
```

Troubleshooting AppInstanceBots configured with Amazon Lex V2 bots

The following topics explain how to troubleshoot common problems with AppInstanceBots.

Finding Amazon Lex V2 failures

The Amazon Chime SDK messaging delivers Amazon EventBridge events when an error prevents it from invoking the Amazon Lex V2 bot. For more information about setting up rules and configuring notification targets, refer to Getting started with Amazon EventBridge in the *Amazon EventBridge User Guide*.

If you receive EventBridge events in AWS CloudWatch Logs, you can use AWS CloudWatch Logs Insights to query EventBridge events based on the Amazon Chime SDK messaging detail-type. The `failureReason` lists the cause of the failure.

The following example shows a typical query.

```
fields @timestamp, @message
| filter `detail-type` = "Chime Messaging AppInstanceBot Lex Failure"
| sort @timestamp desc
```

If Amazon Chime SDK Messaging can invoke your Amazon Lex V2 bot, the SDK sends `CONTROL` messages with an error message.

Troubleshooting Amazon Lex V2 bot permission errors

For an AppInstanceBot to invoke an Amazon Lex V2 Bot, the Amazon Chime SDK messaging service principal must have permission to invoke the Amazon Lex V2 Bot resource. Also, ensure the `AWS:SourceArn` of the resource policy condition matches the ARN of the AppInstanceBot.

For more information about configuring an AppInstanceBot to invoke an Amazon Lex V2 bot, refer to Creating an Amazon Lex V2 bot, earlier in this section.

Troubleshooting Amazon Lex V2 bot throttling

Amazon Lex has a service quota for the maximum number of concurrent text-mode conversations per bot alias. You can contact the Amazon Lex service team for quota increases. For more information, refer to Amazon Lex guidelines and quotas in the *Amazon Lex Developer Guide*.

Managing message retention

Account owners can use the Amazon Chime SDK APIs to turn on retention for messaging. Messages are automatically deleted based on the time period set by the administrator. Retention periods can last from one day to 15 years. You can also use the APIs to update message retention periods or turn off message retention at any time.

Topics in this section

Example CLI retention commands

The following examples show typical CLI commands for retention:

Enabling

```
aws chime-sdk-identity put-app-instance-retention-settings --app-
instance-arn {appInstanceArn} --app-instance-retention-settings
ChannelRetentionSettings={RetentionDays=60}
```

Updating

```
aws chime-sdk-identity put-app-instance-retention-settings --app-
instance-arn {appInstanceArn} --app-instance-retention-settings
ChannelRetentionSettings={RetentionDays=30}
```

Disabling

```
aws chime-sdk-identity put-app-instance-retention-settings --app-
instance-arn {appInstanceArn} --app-instance-retention-settings
ChannelRetentionSettings={}
```

Enabling message retention

You use the Amazon Chime SDK APIs to turn on retention for messaging. You can also use the APIs to update message retention periods or turn off message retention at any time. For more information about configuring messaging retention, see the Amazon Chime SDK API Reference.

Restoring and deleting messages

You can restore messages to users within 30 days of setting or updating a message-retention period. However, after that 30-day grace period, all messages that fall under the retention period are permanently deleted, and new messages are permanently deleted as soon as they pass the retention period.

> **Note**
>
> During the 30-day grace period, if you the extend the retention policy, or you turn it off, messages that haven't passed the new retention period become visible again to the users in the account.

Messages are also permanently deleted when an `AppInstanceUser` deletes a channel or a message.

User interface components for messaging

You can use a component library to reduce the effort needed to build the user interface for chat messaging. See Amazon Chime React Component Library on GitHub for more information.

Integrating with client libraries

To use the messaging features of the Amazon Chime SDK, you must integrate your client application with the following client libraries:

- **AWS SDK** – Contains APIs for sending messages and managing resources.
- **Amazon Chime SDK client library for JavaScript (NPM)** – A JavaScript library with TypeScript type definitions that helps you integrate your client with the Amazon Chime SDK messaging web socket to receive messages.

To integrate your client application with the Amazon Chime SDK, see the instructions in the client library README.md, and use the demos to learn how to learn how to build messaging features.

Using Amazon Chime SDK messaging with JavaScript

You can use JavaScript to manage Amazon Chime SDK resources and send messages. For more information, see the AWS JavaScript SDK.

You can also create a messaging session in your client application to receive messages from the Amazon Chime SDK messaging. For more information, see Using the Amazon Chime SDK client library for JavaScript on GitHub.

Using the Amazon Chime SDK PSTN Audio service

> **Note**
>
> This section describes the Chime SDK PSTN Audio service, which was previously referred to as “SIP Media Applications (SMA)” in prior versions of the documentation and some blog posts. Going forward, when we refer to “SIP Media Applications,” we are referring to the configuration items in the Amazon Chime SDK console and the AWS SDK that are associated with the PSTN Audio service.

This section explains how to use the Amazon Chime SDK Public Switched Telephone Network (PSTN) Audio service. With the PSTN Audio service, developers can build custom telephony applications using the agility and operational simplicity of a serverless AWS Lambda function.

Your AWS Lambda functions control the behavior of phone calls, such as playing voice prompts, collecting digits, recording calls, routing calls to the PSTN and Session Initiation Protocol (SIP) devices using the Amazon Chime SDK Voice Connector. The following topics provide an overview and architectural information about the PSTN Audio service, including how to build AWS Lambda functions to control calls.

> **Note**
>
> The topics in this section assume that you understand the AWS Lambda service. For more information about AWS Lambda, see Getting started with AWS Lambda. Also, to use this section of the Amazon Chime SDK successfully, an Amazon Chime SDK administrator must create at least one SIP rule and one SIP media application. For more information about completing those tasks, see Managing SIP media applications in the *Amazon Chime SDK Administrator Guide*.

Topics

- Routing calls and events to AWS Lambda functions
- Learn about using PSTN Audio service call legs
- Understanding call flow
- Building AWS Lambda functions for the PSTN Audio service

Migrating to the Amazon Chime SDK Voice namespace

The Amazon Chime SDK Voice namespace is a dedicated place for the APIs that create and manage Amazon Chime SDK voice resources. You use the namespace to address Amazon Chime SDK voice API endpoints in any AWS Region that makes them available. If you're just starting to use the Amazon Chime SDK, use this namespace. For more information about Regions, refer to Available AWS Regions for the Amazon Chime SDK service in this guide.

Existing applications that use the Amazon Chime namespace should plan to migrate to the dedicated namespace in order to use updated APIs and new features.

Topics

- Reasons to migrate
- Before you migrate
- Differences between the namespaces

Reasons to migrate

We recommend migrating to the Amazon Chime SDK Voice namespace for the following reasons:

Choice of API endpoint

The Amazon Chime SDK Voice namespace allows you to use API endpoints in any Region that makes them available. If you want to use API endpoints other than `us-east-1`, you must use the Amazon Chime SDK Voice namespace. For more information about the current endpoints, refer to API mapping in this guide.

Updated and new voice APIs

We only add or update voice APIs in the Amazon Chime SDK Voice namespace.

Before you migrate

Before you migrate, be aware of the differences between the namespaces. The following table lists and describes them.

	Amazon Chime SDK Voice namespace	Amazon Chime namespace
AWS namespace	`ChimeSDKVoice`	`Chime`
Regions	Multiple	`us-east-1` only
Endpoints	`https://voice-chime.` *`region`*`.amazonaws.com`	`service.chime.aws.amazon.com`
Service principal	`chime.amazonaws.com`	`chime.amazonaws.com`
APIs	Only APIs for the PSTN Audio service	APIs for PSTN Audio and other parts of Amazon Chime
Voice Connector management	Multiple Regions	`us-east-1`
Voice Connector group management	Multiple Regions	`us-east-1`
SIP media application and SIP rule management	Multiple Regions	`us-east-1`
Phone number management	Multiple Regions	`us-east-1`
Call Analytics	Available	Not available
Voice Profile Domains	Available	Not available
Emergency calling management	Multiple Regions	`us-east-1`

	Amazon Chime SDK Voice namespace	Amazon Chime namespace
Proxy phone session management	Multiple Regions	`us-east-1`
Streaming management	Multiple Regions	`us-east-1`
Logging and metrics management	Multiple Regions	`us-east-1`

For more information about the available Regions, refer to Voice Regions.

Differences between the namespaces

The following sections explain the differences between the `Amazon Chime SDK Voice` and `Amazon Chime` namespaces.

AWS namespace

The Amazon Chime SDK namespace uses the `Chime` formal name. The Amazon Chime SDK Voice namespace uses the `ChimeSDKVoice` formal name. The precise format of the name varies by platform.

For example, if you use the AWS SDK in Node.js to create meetings, you use the following line of code to address the namespace.

```
const chimeVoice = AWS.Chime();
```

To migrate to the Amazon Chime SDK Voice namespace, update this line of code with the new namespace and the endpoint region.

```
const chimeVoice = AWS.ChimeSDKVoice({ region: "eu-central-1" });
```

Regions

The Amazon Chime namespace can only address API endpoints in the `us-east-1` Region. The Amazon Chime SDK Voice namespace can address Amazon Chime SDK voice API endpoints in any

Region they are available. For a current list of voice Regions, refer to Available AWS Regions for the Amazon Chime SDK service in this guide.

Endpoints

The Amazon Chime SDK Voice namespace uses different API endpoints than the Amazon Chime namespace.

Only the endpoint used to create a voice action can be used to modify it. This means a voice action created via an endpoint in `eu-central-1` can only be modified via `eu-central-1`. It also means you cannot address a voice action created via the `Chime` namespace with the `ChimeSDKVoice` namespace in `us-east-1`. For more information about the current endpoints, refer to API mapping in this guide.

Service principal

Both namespaces uses the `chime.amazonaws.com` service principal. If you have access policies that grant access to the service, you do not need to update those polices.

APIs

The Amazon Chime SDK Voice namespace only contains APIs to create and manage voice actions. The Amazon Chime namespace includes APIs for voice and other parts of the Amazon Chime service, such as meetings.

Tagging

Only the Amazon Chime SDK Voice namespace supports tags. For more information about tags, refer to TagResource and UntagResource.

Media Regions

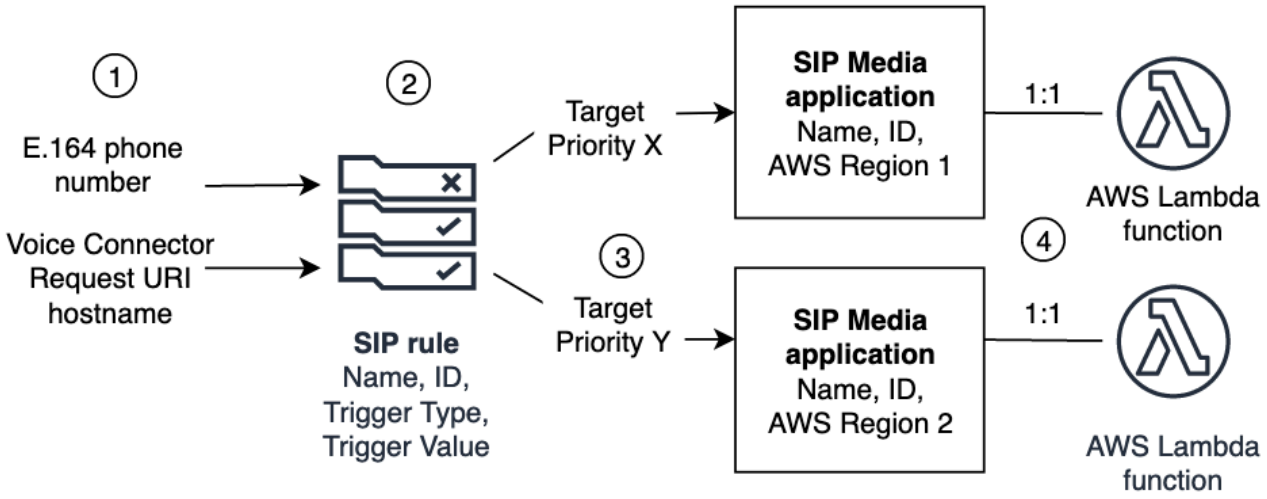
If you use the `Chime` namespace to create resources such as Voice Connectors and SIP media application, you can only do so in `us-east-1` and `us-west-2`, and you must use the `us-east-1` endpoint.

In contrast, the `ChimeSDKVoice` namespace allows resource creation in all supported Regions, not just `us-east-1` and `us-west-2`. For more information about regional coverage, refer to Available AWS Regions for the Amazon Chime SDK service.

Understanding phone numbers, SIP rules, SIP media applications, and AWS Lambda functions

Before you can use the PSTN Audio service, an Amazon Chime SDK administrator must provision your phone numbers and create managed objects called SIP rules and SIP media applications. You can use the Amazon Chime SDK console or the AWS SDK to provision phone numbers, and to provision the SIP rule and SIP media application managed objects.

This image shows the relationship between the managed objects that comprise the PSTN Audio service. Numbers in the image correspond to numbers in the text below the image.



You can only assign phone numbers and Amazon Chime SDK Voice Connectors (1) to SIP rules (2). Also, you must provision the phone number or Voice Connector in your PSTN Audio service. Upon receiving an inbound call to a phone number, or an outbound call request from a Voice Connector, the SIP rule invokes a SIP media application and an associated AWS Lambda function (4). The AWS Lambda function runs a predefined set of actions, such as playing on-hold music or joining a meeting. To provide multi-region resiliency, SIP rules can specify alternate target SIP media applications in different AWS Regions (3) by order of priority for failover. If one target fails, the PSTN Audio service tries the next one and so on. Note that each alternate target must reside in a different AWS Region.

In addition, multiple SIP media applications can invoke a given AWS Lambda function. Put another way, when you create an AWS Lambda function, any SIP media application can use that function.

For more information about provisioning SIP media applications and rules, see Managing SIP media applications and rules in the *Amazon Chime SDK Administrator Guide*.

Understanding the PSTN Audio service programming model

The PSTN Audio service uses a request/response programming model that in turn uses AWS Lambda functions. Your AWS Lambda function is invoked automatically for incoming and outgoing calls. For example, when a new incoming call arrives, the PSTN Audio service invokes your AWS Lambda function with a `NEW_INCOMING_CALL` event and waits for commands called *Actions*. For example, your application can choose actions such as playing an audio prompt, collecting digits, recording audio, or routing the call onward. These JSON formatted actions are sent back to the PSTN Audio service using a callback from your AWS Lambda function.

This example shows a `PlayAudio` action.

```
{
    "Type": "PlayAudio",
    "Parameters": {
        "CallId": "call-id-1",
        "ParticipantTag": "LEG-A",
        "PlaybackTerminators": ["1", "8", "#"],
        "Repeat": "5",
        "AudioSource": {
            "Type": "S3",
            "BucketName": "valid-S3-bucket-name",
            "Key": "wave-file.wav"
        }
    }
}
```

This example shows a `RecordAudio` action.

```
{
    "Type": "RecordAudio",
    "Parameters": {
        "CallId": "call-id-1",
        "DurationInSeconds": "10",
        "SilenceDurationInSeconds": 3,
        "SilenceThreshold": 100,
        "RecordingTerminators": [
            "#"
```

```
        ],
        "RecordingDestination": {
            "Type": "S3",
            "BucketName": "valid-bucket-name",
            "Prefix": "valid-prefix-name"
        }
    }
}
```

Once the PSTN Audio service runs the action, it invokes your AWS Lambda function again with either a success or failure indication.

Your application can also make outbound phone calls and use your AWS Lambda function to control the call flow, caller experience, and call context. In this case, you call the CreateSipMediaApplicationCall API, and your AWS Lambda is invoked with a `NEW_OUTBOUND_CALL` event. Once the call is answered, you can return actions, such as playing a voice prompt and collecting user-entered digits. You can also trigger your AWS Lambda function using the UpdateSipMediaApplicationCall API to implement timers, participant muting, and waiting rooms.

Routing calls and events to AWS Lambda functions

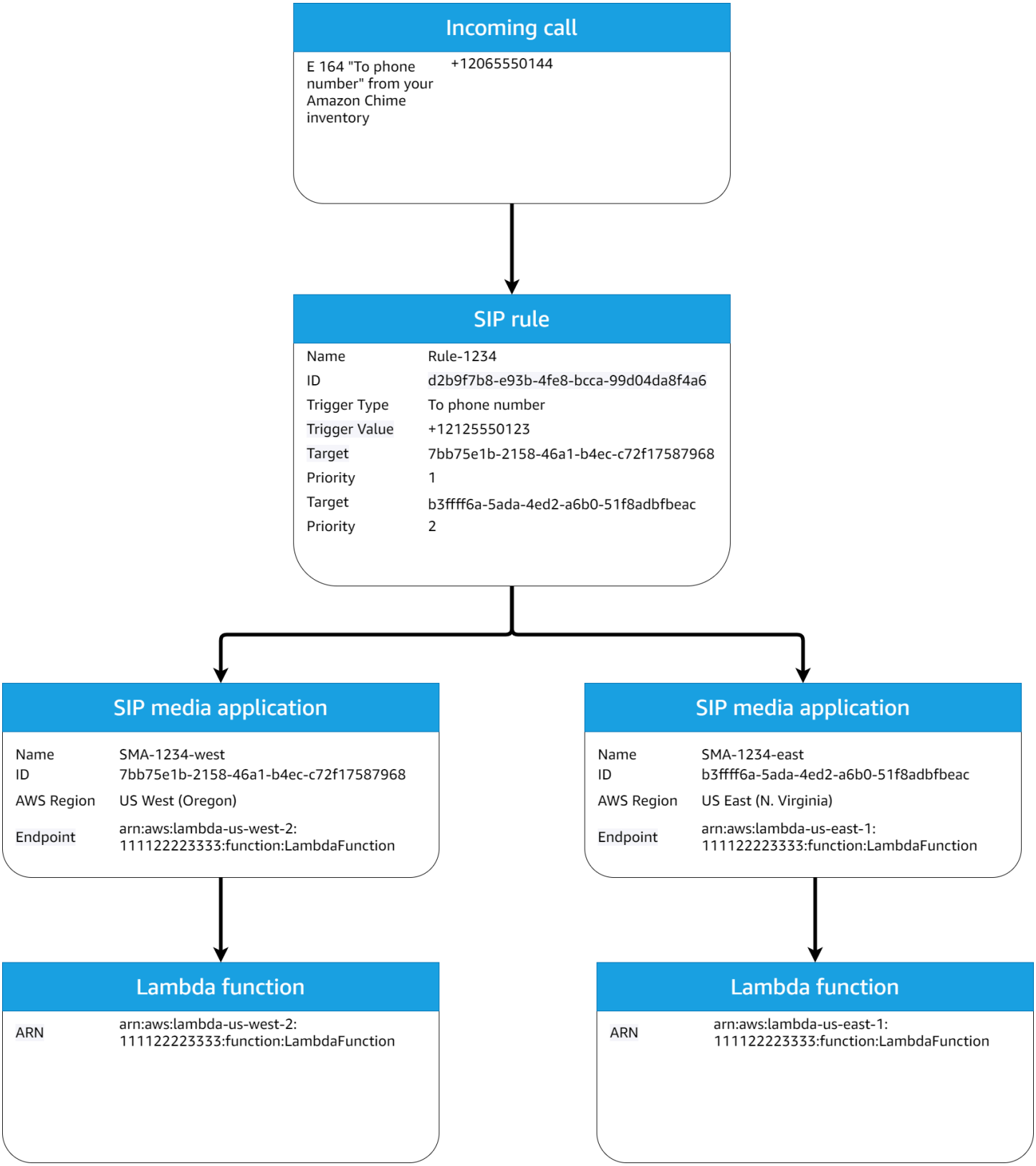
The PSTN Audio service provides the following ways to route incoming phone calls to your AWS Lambda function for treatment.

- You can route calls based on the called number. To do this, an Amazon Chime SDK administrator creates a SIP rule with the **Trigger Type** set to **To phone number**. This phone number must exist in the Amazon Chime SDK phone number inventory, in the same AWS account as the SIP rule.
- You can route calls to the AWS Lambda function based on the request URI of an incoming Voice Connector SIP call. To do this, an Amazon Chime SDK administrator creates a SIP rule with the **Trigger Type** set to **Request URI hostname**. This field must contain a fully-qualified domain name specified in the “outbound host name” field of a Voice Connector that is provisioned in the same AWS account as the SIP rule.

Next, the administrator provisions at least one target SIP media application. Optionally, you can provision multiple SIP media applications in priority order to support redundancy and failover. For example, you can provision two SIP media applications in two different AWS regions and specify their order of priority. If a SIP rule has more than one target SIP media application, the SIP media application’s Lambda functions are invoked in the order of priority. The AWS Lambda function in

the SIP media application with the highest order of priority (the smallest number, such as 1) runs first. If the PSTN Audio service can't invoke that AWS Lambda function, the AWS Lambda function in the SIP media application with the next highest order of priority (the next least number, such as 2) is invoked. If all attempts to run the SIP media applications specified in the SIP rule fail, the PSTN Audio service hangs up.

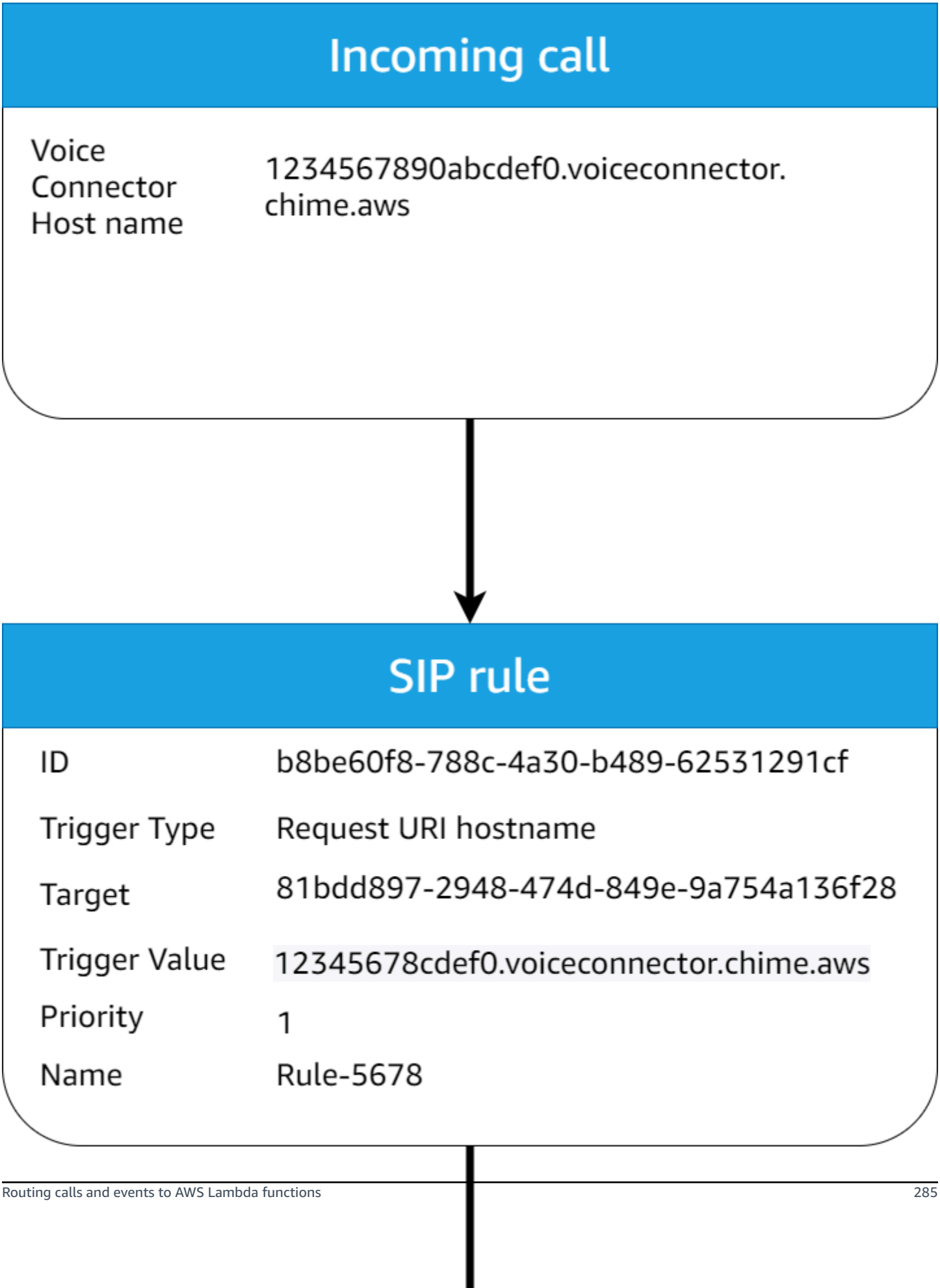
Once the necessary SIP rules and SIP media applications are provisioned, the PSTN Audio service routes incoming calls to your AWS Lambda function. The following diagram shows a typical sequence using the **To phone number** trigger type.



In the diagram:

1. The PSTN Audio service receives an incoming call to a phone number that is provisioned in a SIP rule in the same AWS account.
2. The PSTN Audio service then evaluates the SIP rule and fetches the SIP media application with the highest order of priority (in this case, priority 1).
3. The service then invokes the AWS Lambda function associated with the SIP media application.
4. Optional. If the service can't invoke the associated AWS Lambda with the highest order of priority, it will attempt to run the SIP media application with the next highest order of priority (in this case, priority 2), if one exists.
5. Optional. If all the target SIP media applications fail, the PSTN Audio service hangs up the call.

The following diagram shows a typical rule that uses a **Request URI hostname** trigger type.


Incoming call
Voice Connector Host name
1234567890abcdef0.voiceconnector.chime.aws
SIP rule
ID
b8be60f8-788c-4a30-b489-62531291cf
Trigger Type
Request URI hostname
Target
81bdd897-2948-474d-849e-9a754a136f28
Trigger Value
12345678cdef0.voiceconnector.chime.aws
Priority
1
Name
Rule-5678

In the diagram:

1. The PSTN audio service receives an incoming call on an Amazon Chime SDK Voice Connector with a **Request URI hostname** that matches a provisioned SIP rule in the same AWS account.
2. The service then evaluates the SIP rule and fetches the SIP media application with the lowest priority (in this case, the only target SIP media application with priority 1).
3. The service then invokes the AWS Lambda function associated with the SIP media application.
4. Optional. If the service cannot invoke the associated AWS Lambda with the lowest priority, it tries to run the SIP media application with the next lowest priority, if one exists. In this case, there is only one target SIP media application.
5. Optional. If all the target SIP media applications fail, the PSTN Audio service hangs up the call.

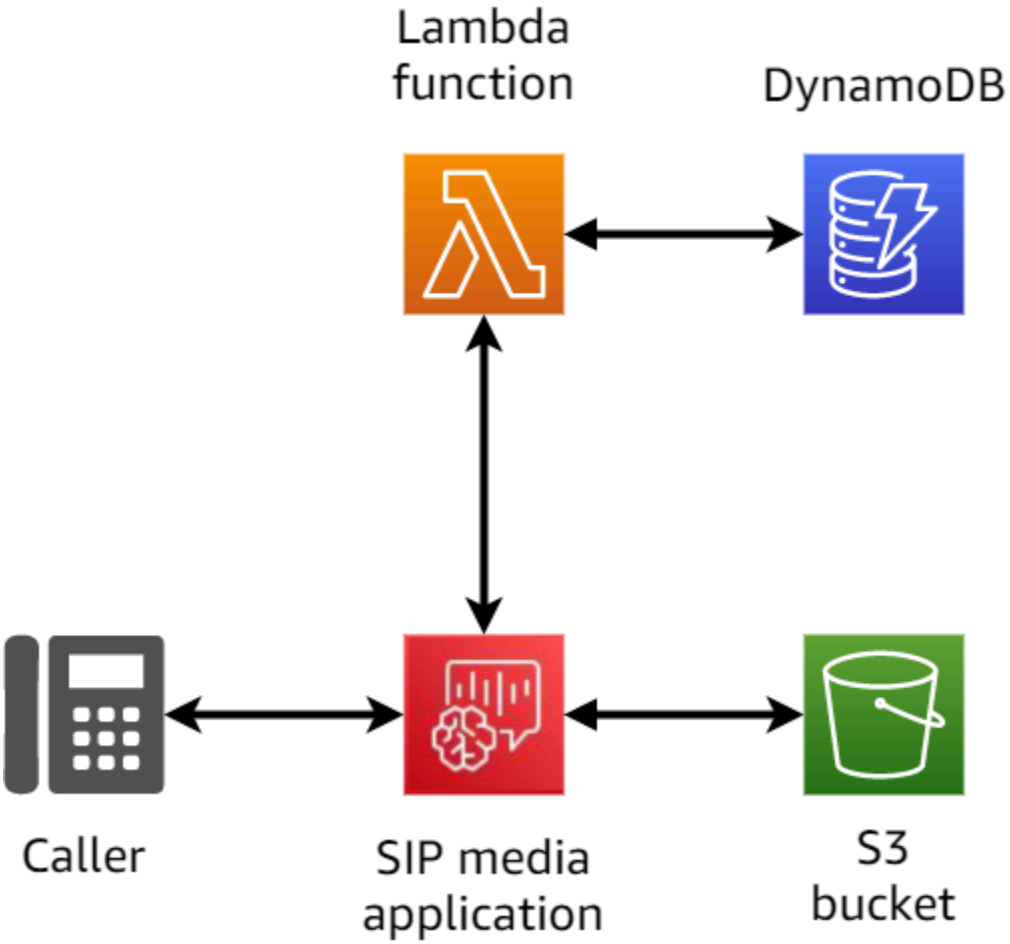
Additionally, you can create an outbound call, and subsequently invoke your AWS Lambda function for additional processing, using the CreateSIPMediaApplicationCall API. To use this API, you specify the provisioned **SIP media application ID** as a parameter.

Finally, you can trigger your AWS Lambda function at any time while a call is active using the UpdateSIPMediaApplicationCall API. To use the API, you specify the provisioned **SIP media application ID** as a parameter.

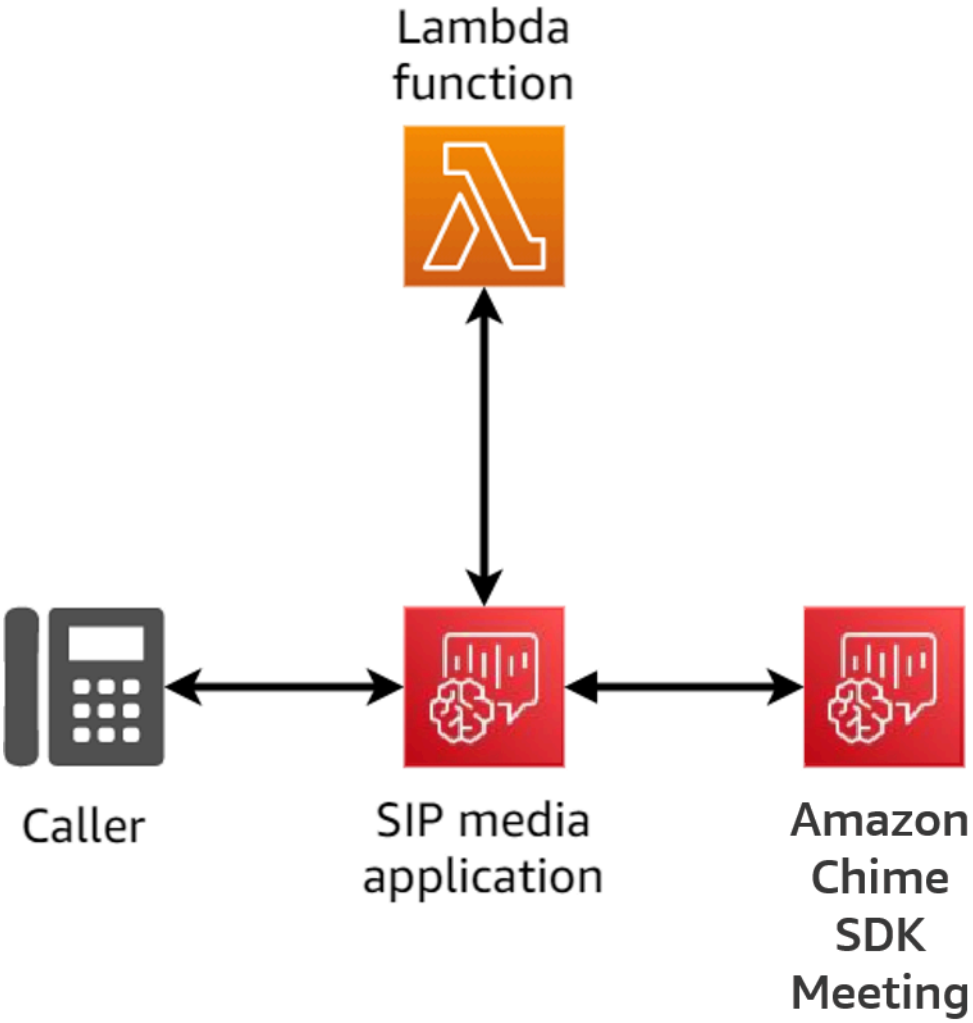
Learn about using PSTN Audio service call legs

The PSTN Audio service can operate on one or more call legs. For example, you have a single call leg when you record or deliver a voice mail, and you have multiple call legs when you join an Amazon Chime SDK meeting.

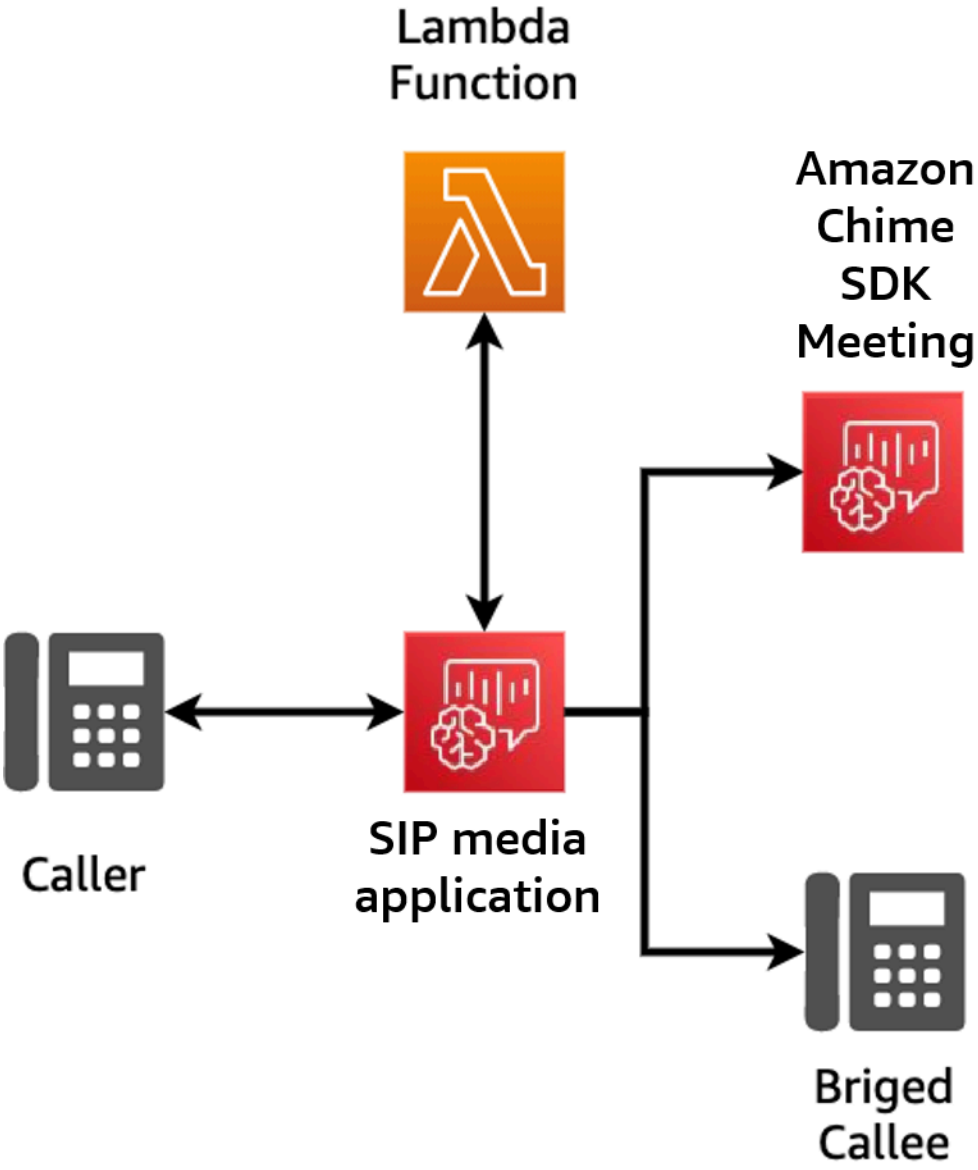
The following diagram shows the flow of a single-leg call.



The following diagram shows the architecture of a multi-leg call.



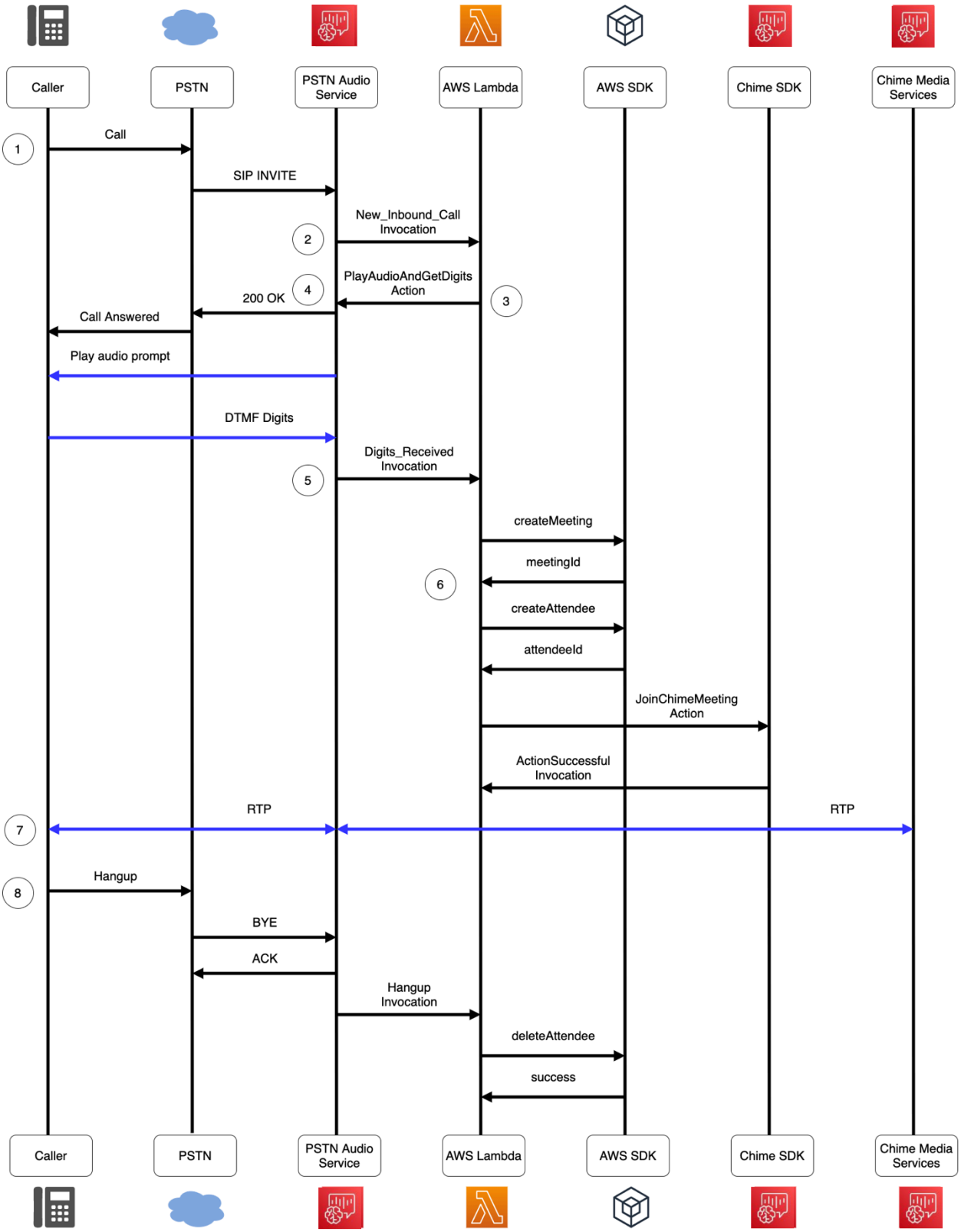
The following diagram shows the flow of a multi-leg bridged call.



Understanding call flow

This diagram shows the flow of a call through the Amazon Chime SDK PSTN Audio service and a customer's AWS Lambda function. In this example, the application plays a prompt to the caller, gathers dual-tone multi frequency (DTMF) digits, and then connects them to an Amazon Chime SDK meeting.

Numbers in the diagram correspond to the numbered explanations below the diagram.



In the diagram:

1. The Amazon Chime SDK PSTN audio service receives a call to a phone number that is provisioned in a SIP rule.
2. The PSTN audio service fetches the associated SIP media application and invokes the associated AWS Lambda function with a `NEW_INBOUND_CALL` event (LEG-A).
3. The AWS Lambda function returns a list of actions, including `PlayAudioAndGetDigits`, which instructs the PSTN Audio service to answer the call, play an audio file to the caller, and collect the DTMF digits entered by the caller.
4. The PSTN Audio service answers the call, plays an audio prompt, and collects DTMF digits input by the caller.
5. The PSTN Audio service invokes the AWS Lambda function with the DTMF digits input. The AWS Lambda function uses the AWS SDK to create an Amazon Chime SDK meeting and a meeting attendee.
6. Once the AWS SDK returns a `MeetingId` and `AttendeeId`, the AWS Lambda function returns an action to join the call to the Amazon Chime SDK Meeting (LEG-B).
7. A Real-time Transport Protocol (RTP) session is established between the caller from the public switched telephone network (PSTN) and the Amazon Chime SDK Media service.
8. When the PSTN caller hangs up, the PSTN Audio service invokes the AWS Lambda function with a HANGUP event, and the AWS Lambda function deletes the attendee.

Building AWS Lambda functions for the PSTN Audio service

The topics in this section explain how to build the AWS Lambda functions used by your PSTN Audio service.

Contents

- Understanding timeouts and retries
- Debugging and troubleshooting
- Understanding VoiceFocus
- PSTN audio service glossary

Understanding telephony events

The Audio Service invokes your AWS AWS Lambda function when certain events occur during a call. The following example shows the events, and text after the example explains each event.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 3,
    "InvocationEventType": "event-type",
    "CallDetails": {
        "TransactionId": "transaction-id-1",
        "AwsAccountId": "aws-acct-id-1",
        "AwsRegion": "us-east-1",
        "SipMediaApplicationId": "sip-media-app-id-1",
        "Participants": [
            {
                "CallId": "call-id-1",
                "ParticipantTag": "LEG-A",
                "To": "e164PhoneNumber",
                "From": "e164PhoneNumber",
                "Direction": "Inbound/Outbound",
                "StartTimeInMilliseconds": "1641998241509",
                "Status": "Connected/Disconnected"
            }
        ]
    }
}
```

SchemaVersion

The version of schema used to create this event object.

Sequence

The sequence of events that invoke your AWS Lambda function. Each time your function is invoked during a call, the sequence is incremented.

InvocationEventType

The type of event that triggers an AWS Lambda invocation. For more information, see Event types later in this topic.

CallDetails

Information about the call associated with the AWS Lambda invocation.

TransactionId

The ID of a call associated with an AWS Lambda invocation.

AwsAccountId

The AWS account ID associated with the SIP media application that resulted in the call routing.

SipMediaApplicationId

The ID of the SIP media application associated with the call.

Participants

Information about the participants on the call that invokes an AWS AWS Lambda function.

CallId

A unique ID assigned to each participant.

ParticipantTag

Each call participant gets a tag, `LEG-A` or `LEG-B`.

To

The participant "to" phone number, in E.164 format.

From

The participant “from” phone number, in E.164 format.

Direction

The direction that a call leg comes from. `Inbound` represents a call made to the Audio Service. `Outbound` represents a call made from the Audio Service.

StartTimeInMilliseconds

The epoch time in milliseconds, starting when a participant joins a call.

Status

Whether a participant is `Connected` or `Disconnected`

Event types

The Audio Service invokes the Lambda function with these event types:

NEW_INBOUND_CALL

A new call has been initiated by a phone number associated with your SIP media application.

NEW_OUTBOUND_CALL

A new outbound call has been made via the CreateSipMediaApplicationCall API.

ACTION_SUCCESSFUL

An action returned from your AWS Lambda function has succeeded. Successful actions include `ActionData` that matches the successful action.

```
"ActionData": {
    // The previous successful action
},
```

ACTION_FAILED

An action returned from your AWS Lambda function did not succeed. Unsuccessful actions include `ActionData` that matches the failed action, an error type, and an error message that describes the failure:

```
"ActionData": {
    // The previous unsuccessful action
    "ErrorType": "error-type",
    "ErrorMessage": "error message"
},
```

ACTION_INTERRUPTED

An action in the process of running was interrupted by an UpdateSipMediaApplicationCall API invocation. The `ActionData` includes the interrupted actions:

```
"ActionData": {
        // The action that was interrupted
    },
```

HANGUP

A user or the application hung up a call leg. The `ActionData` includes these details about the event:

```
  "ActionData": {
       "Type": "Hangup",
       "Parameters": {
           "SipResponseCode": 486,
           "CallId": "c70f341a-adde-4406-9dea-1e01d34d033d",
           "ParticipantTag": "LEG-A"
       }
    },
```

Type

Hangup.

Parameters

The information about the HANGUP event:

- **SipResponseCode** – The response code associated with the event. The most common codes are:
  - **0** – Normal clearing
  - **480** – No answer
  - **486** – User busy
- **CallId** The ID of the participant that hung up.
- **ParticipantTag** The tag of the participant that hung up.

CALL_ANSWERED

The Audio Service answered an incoming call was answered. This event is returned on a dial-out call unless the call is bridged.

INVALID_LAMBDA_RESPONSE

The response provided to the last AWS Lambda invocation caused a problem. The `ActionData` includes these additional fields:

```
"ErrorType": "error-type-1",
"ErrorMessage": "error-msg-1"
```

DIGITS_RECEIVED

The application received DTMF digits after completion of a `ReceiveDigits` action. The `ActionData` includes the received digits.

```
"ActionData": {
    "ReceivedDigits": ###
    // The ReceiveDigits action data
},
```

CALL_UPDATE_REQUESTED

The UpdateSipMediaApplicationCall API was invoked. The `ActionData` includes information about the update request:

```
"ActionData": {
    "Type": "CallUpdateRequest",
    "Parameters": {
        "Arguments": {
            "leg": "LEG-A"
            }
        }
    },
}
```

RINGING

A call leg is ringing

Understanding PSTN Audio service actions

In the PSTN Audio service, SIP media applications trigger AWS Lambda functions. In turn, the AWS Lambda functions can return a list of instructions known as *actions*. An action is an item that you want to run on a leg of a phone call, such as sending or receiving digits, joining a meeting, and so on. Actions can also return data, so you can think of actions as objects with data fields. For more information about the actions invoked by the PSTN Audio service, see Understanding telephony events.

Learn about the telephony events that invoke AWS Lambda functions

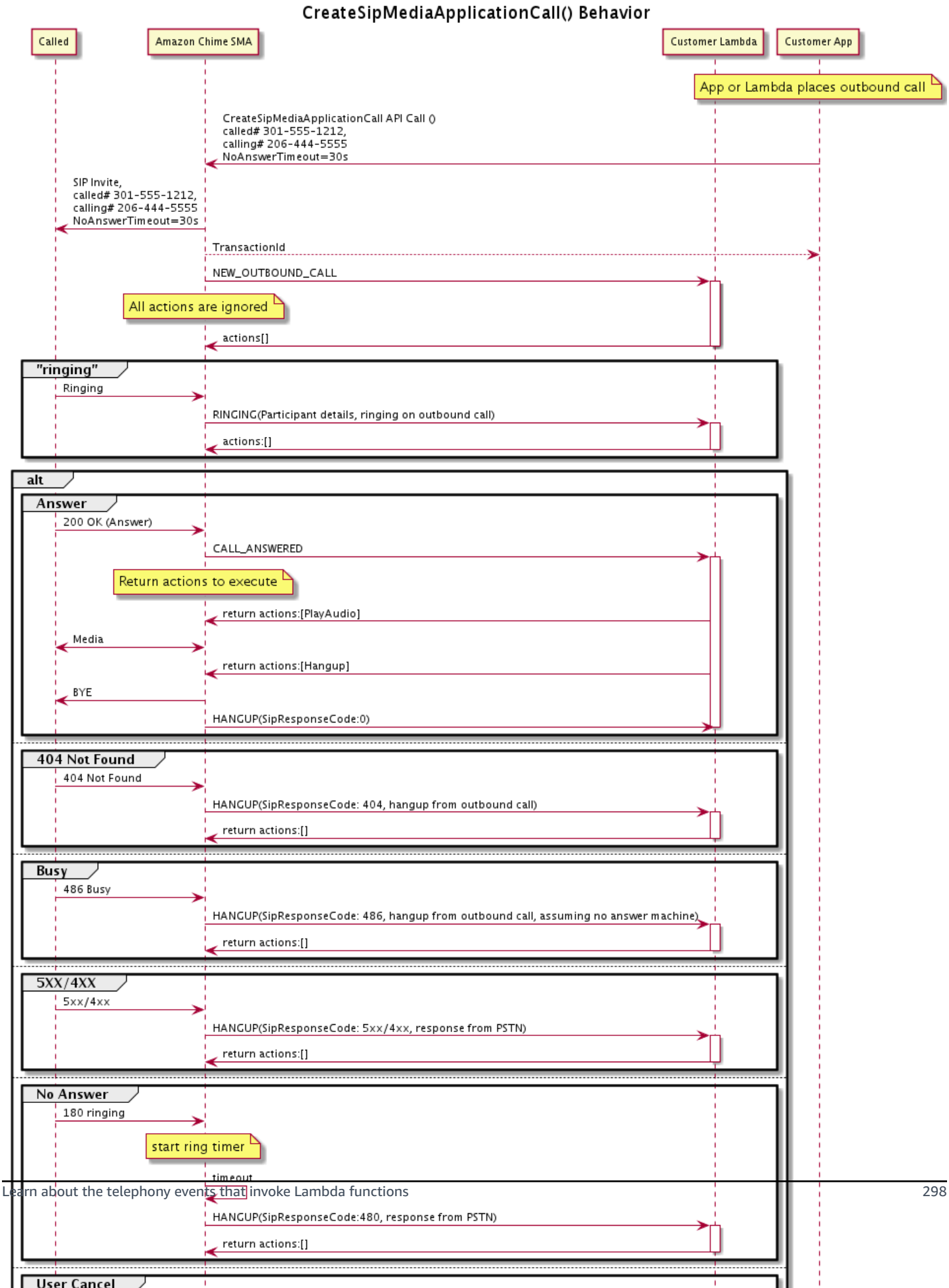
The Audio Service invokes AWS Lambda functions in response to different events. Each invocation specifies an invocation event type and provides the call details, including its participants, if applicable. The following topics describe the Audio Service events that invoke AWS Lambda functions.

Making an outbound call

To create an outbound call, you use the CreateSipMediaApplicationCall API. The API invokes the endpoint of a specified `SIP media application ID`. Customers can control the flow of the call by giving different signaling and SipMediaApplication actions from the endpoint.

In the event of a successful response, the API returns a 202 http status code along with a transactionId, which you can use with the UpdateSipMediaApplicationCall API to update an in-progress call.

The following diagram shows the invocations made to the AWS Lambda function endpoint for an outbound call.


CreateSipMediaApplicationCall() Behavior
Called
Amazon Chime SMA
Customer Lambda
Customer App
App or Lambda places outbound call
CreateSipMediaApplicationCall API Call ()
called# 301-555-1212,
calling# 206-444-5555
NoAnswerTimeout=30s
SIP Invite,
called# 301-555-1212,
calling# 206-444-5555
NoAnswerTimeout=30s
TransactionId
NEW_OUTBOUND_CALL
All actions are ignored
actions[]
"ringing"
Ringing
RINGING(Participant details, ringing on outbound call)
actions:[]
alt
Answer
200 OK (Answer)
CALL_ANSWERED
Return actions to execute
return actions:[PlayAudio]
Media
return actions:[Hangup]
BYE
HANGUP(SipResponseCode:0)
404 Not Found
404 Not Found
HANGUP(SipResponseCode: 404, hangup from outbound call)
return actions:[]
Busy
486 Busy
HANGUP(SipResponseCode: 486, hangup from outbound call, assuming no answer machine)
return actions:[]
5XX/4XX
5xx/4xx
HANGUP(SipResponseCode: 5xx/4xx, response from PSTN)
return actions:[]
No Answer
180 ringing
start ring timer
timeout
HANGUP(SipResponseCode:480, response from PSTN)
return actions:[]
User Cancel

The endpoint configured for the SIP media application is invoked for different statuses of the outbound call. When a customer initiates a call, The Amazon Chime SDK invokes the endpoint with a `NEW_OUTBOUND_CALL` invocation event type.

This example shows a typical invocation event for a `NEW_OUTBOUND_CALL`.

```
{
    "SchemaVersion": "1.0",
        "Sequence": 1,
        "InvocationEventType": "NEW_OUTBOUND_CALL",
        "CallDetails": {
            "TransactionId": "transaction-id",
            "AwsAccountId": "aws-account-id",
            "AwsRegion": "us-east-1",
            "SipApplicationId": "sip-application-id",
            "Participants": [
                {
                    "CallId": "call-id-1",
                    "ParticipantTag": "LEG-A",
                    "To": "+1xxxx",
                    "From": "+1xxxxxxx",
                    "Direction": "Outbound",
                    "StartTimeInMilliseconds": "159700958834234"
                }
            ]
    }
}
```

Any response for an event related AWS Lambda invocation is ignored.

When we receive a `RINGING` notification from the receiver, the Amazon Chime SDK invokes the configured endpoint again.

This example shows a typical invocation event for `RINGING`.

```
{
    "SchemaVersion": "1.0",
        "Sequence": 1,
        "InvocationEventType": "RINGING",
        "CallDetails": {
            "TransactionId": "transaction-id",
            "AwsAccountId": "aws-account-id",
            "AwsRegion": "us-east-1",
```

```
            "SipApplicationId": "sip-application-id",
            "Participants": [
                {
                    "CallId": "call-id-1",
                    "ParticipantTag": "LEG-A",
                    "To": "+1xxxx",
                    "From": "+1xxxxxxx",
                    "Direction": "Outbound",
                    "StartTimeInMilliseconds": "159700958834234"
                }
            ]
    }
}
```

Any response for an event related AWS Lambda invocation is ignored.

If the receiver doesn't answer the call, or the call fails due to an error, Chime disconnects the call and invokes the endpoint with the `Hangup` event type. For more information about the `Hangup` event type, refer to Ending a call.

If the call is answered, Chime invokes the endpoint with the `CALL_ANSWERED` action. This example shows a typical invocation event.

```
{
  "SchemaVersion": "1.0",
    "Sequence": 1,
    "InvocationEventType": "CALL_ANSWERED",
    "CallDetails": {
        ""TransactionId": "transaction-id",
            "AwsAccountId": "aws-account-id",
            "AwsRegion": "us-east-1",
            "SipApplicationId": "sip-application-id",
            "Participants": [
                {
                    "CallId": "call-id-1",
                    "ParticipantTag": "LEG-A",
                    "To": "+1xxxx",
                    "From": "+1xxxxxxx",
                    "Direction": "Outbound",
                    "StartTimeInMilliseconds": "159700958834234",
                "Status": "Connected"
            }
        ]
```

```
    }
}
```

At this point, you can return actions by responding to the invocation with an action list. If you don't want to run any actions, respond with an empty list. You can respond with a maximum of 10 actions for each AWS Lambda invocation, and you can invoke a Lambda function 1,000 times per call. For more information about responding with sets of actions, refer to Responding to invocations with action lists.

Receiving an inbound call

When a `NEW_INCOMING_CALL` event occurs, the Audio Service creates a unique `TransactionID` and unique `CallID` that persist until the `HANGUP` event occurs.

You can respond in several ways to a `NEW_INCOMING_CALL` event. For example:

- Send `PlayAudio` or `RecordAudio` actions and automatically answer the call.
- Send a `Pause` action.
- Send a `Hangup` action, in which case the call isn't answered and the customer isn't charged.
- Send a `CallAndBridge` action and add another user to the call.
- Do nothing, the call attempt times out after 30 seconds.

When a new inbound call is received, the SIP media application invokes an AWS Lambda function with this payload.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 2,
    "InvocationEventType": "NEW_INBOUND_CALL"
    "CallDetails": {
        "TransactionId": "transaction-id",
        "AwsAccountId": "aws-account-id",
        "AwsRegion": "us-east-1",
        "SipRuleId": "sip-rule-id",
        "SipApplicationId": "sip-application-id",
        "Participants": [
            {
                "CallId": "call-id-1",
                "ParticipantTag": "LEG-A",
```

```
                "To": "+12065551212",
                "From": "+15105550101",
                "Direction": "Inbound",
                "StartTimeInMilliseconds": "159700958834234",
                "Status": "Connected"
            }
        ]
    }
}
```

Specifying actions in response to telephony events

In the Audio Service, SIP media applications invoke AWS Lambda functions. In turn, a Lambda function can return a list of instructions known as *actions*. An action is an item that you want to run on a leg of a phone call, such as sending or receiving digits, joining a meeting, and so on. For more information about the actions invoked by the PSTN Audio service, see Understanding telephony events.

When a SIP media application successfully runs a list of actions, the application calls the AWS Lambda function with an invocation event type of `ACTION_SUCCESSFUL`. If any of the actions fail to complete, the SIP media application calls the AWS Lambda function with the `ACTION_FAILED` event.

The SIP media application only returns `ACTION_SUCCESSFUL` if all the actions on the list succeed. If any of the actions in the list fail, the SIP media application invokes the AWS Lambda function with the `ACTION_FAILED` event and clears the remaining actions in the list after the failed one. Then the SIP media application runs the next action returned by the AWS Lambda function. You use the `ActionData` key to identify which call invoked the function.

The following event shows a sample payload for the `ACTION_SUCCESSFUL` invocation event type after a `PlayAudioAndGetDigits` action.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 3,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type": "PlayAudioAndGetDigits",
        "Parameters" : {
            "CallId": "call-id-1",
            "AudioSource": {
```

```
                "Type": "S3",
                "BucketName": "bucket-name",
                "Key": "failure-audio-file.wav"
            },
            "FailureAudioSource": {
                "Type": "S3",
                "BucketName": "bucket-name",
                "Key": "failure-audio-file.wav"
            },
            "MinNumberOfDigits": 3,
            "MaxNumberOfDigits": 5,
            "TerminatorDigits": ["#"],
            "InBetweenDigitsDurationInMilliseconds": 5000,
            "Repeat": 3,
            "RepeatDurationInMilliseconds": 10000
        },
        "ReceivedDigits": "123"
    }
    "CallDetails": {
        "TransactionId": "transaction-id",
        "AwsAccountId": "aws-account-id",
        "AwsRegion": "us-east-1",
        "SipRuleId": "sip-rule-id",
        "SipApplicationId": "sip-application-id",
        "Participants": [
            {
                "CallId": "call-id-1",
                "ParticipantTag": "LEG-A",
                "To": "+12065551212",
                "From": "+15105550101",
                "Direction": "Inbound",
                "StartTimeInMilliseconds": "159700958834234",
                "Status": "Connected"
                }
            ]
        }
    }
}
```

When any action in a list fails to complete successfully, the SIP media application invokes the AWS Lambda function to notify you of the failure, and to get a new set of actions to run on that call. The following event shows the sample payload for the `ACTION_FAILED` invocation event type after a `PlayAudio` action.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 4,
    "InvocationEventType": "ACTION_FAILED",
    "ActionData": {
        "Type": "PlayAudio",
        "Parameters" : {
            "CallId": "call-id-1",
            "AudioSource": {
                "Type": "S3",
                "BucketName": "bucket-name",
                "Key": "audio-file.wav"
            }
        },
        "ErrorType": "InvalidAudioSource",
        "ErrorMessage": "Audio Source parameter value is invalid."
    }
    "CallDetails": {
        "TransactionId": "transaction-id",
        "AwsAccountId": "aws-account-id",
        "AwsRegion": "us-east-1",
        "SipRuleId": "sip-rule-id",
        "SipApplicationId": "sip-application-id",
        "Participants": [
            {
                "CallId": "call-id-1",
                "ParticipantTag": "LEG-A",
                "To": "+12065551212",
                "From": "+15105550101",
                "Direction": "Inbound",
                "StartTimeInMilliseconds": "159700958834234",
                "Status": "Connected"
            }
        ]
    }
}
}
```

Receiving caller input

You use the `ReceiveDigits` action to collect inbound DTMF digits and match them against a regular expression. When the SIP media application receives digits that match the regular

expression, it invokes a AWS Lambda function with an `ACTION_SUCCESSFUL` event. The collected digits appear in the `ReceivedDigits` value in the `ActionData` object.

For example:

```
{
    "SchemaVersion": "1.0",
    "Sequence": 4,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "ReceivedDigits": "",
        "Type": "ReceiveDigits",
        "Parameters": {
            "CallId": "call-id-1",
            "InputDigitsRegex": "^\d{2}#$",
            "InBetweenDigitsDurationInMilliseconds": 5000,
            "FlushDigitsDurationInMilliseconds": 10000
        }
    },
    "CallDetails": {
        "TransactionId": "transaction-id",
        "AwsAccountId": "aws-account-id",
        "AwsRegion": "us-east-1",
        "SipRuleId": "sip-rule-id",
        "SipApplicationId": "sip-application-id",
        "Participants": [
            {
                "CallId": "call-id-1",
                "ParticipantTag": "LEG-A",
                "To": "+12065551212",
                "From": "+15105550101",
                "Direction": "Inbound",
                "StartTimeInMilliseconds": "159700958834234",
                "Status": "Connected"
            }
        ]
    }
}
```

Once the caller enters digits that match your regular expression pattern, the SIP media application invokes an AWS Lambda function that returns the following type of payload:

```
{
```

```
    "SchemaVersion": "1.0",
    "Sequence": 5,
    "InvocationEventType": "DIGITS_RECEIVED",
    "ActionData": {
        "ReceivedDigits": "11#",
        "Type": "ReceiveDigits",
        "Parameters": {
            "CallId": "call-id-1",
            "InputDigitsRegex": "^\d{2}#$",
            "InBetweenDigitsDurationInMilliseconds": 5000,
            "FlushDigitsDurationInMilliseconds": 10000
        }
    },
    "CallDetails": {
        "TransactionId": "transaction-id",
        "AwsAccountId": "aws-account-id",
        "AwsRegion": "us-east-1",
        "SipRuleId": "sip-rule-id",
        "SipApplicationId": "sip-application-id",
        "Participants": [
            {
                "CallId": "call-id-1",
                "ParticipantTag": "LEG-A",
                "To": "+12065551212",
                "From": "+15105550101",
                "Direction": "Inbound",
                "StartTimeInMilliseconds": "159700958834234",
                "Status": "Connected"
            }
        ]
    }
}
```

See a working example on GitHub: https://github.com/aws-samples/amazon-chime-sma-on-demand-recording

Updating in-progress calls

As part of the PSTN Audio Service, SIP media applications allow you to set actions that are run on a call by invoking user-defined Lambda functions based on the call events, such as an incoming call or DTMF digits. The UpdateSipMediaApplicationCall API allows you to trigger a Lambda function at any time while a call is active, replacing the current actions with new actions returned by the invocation.

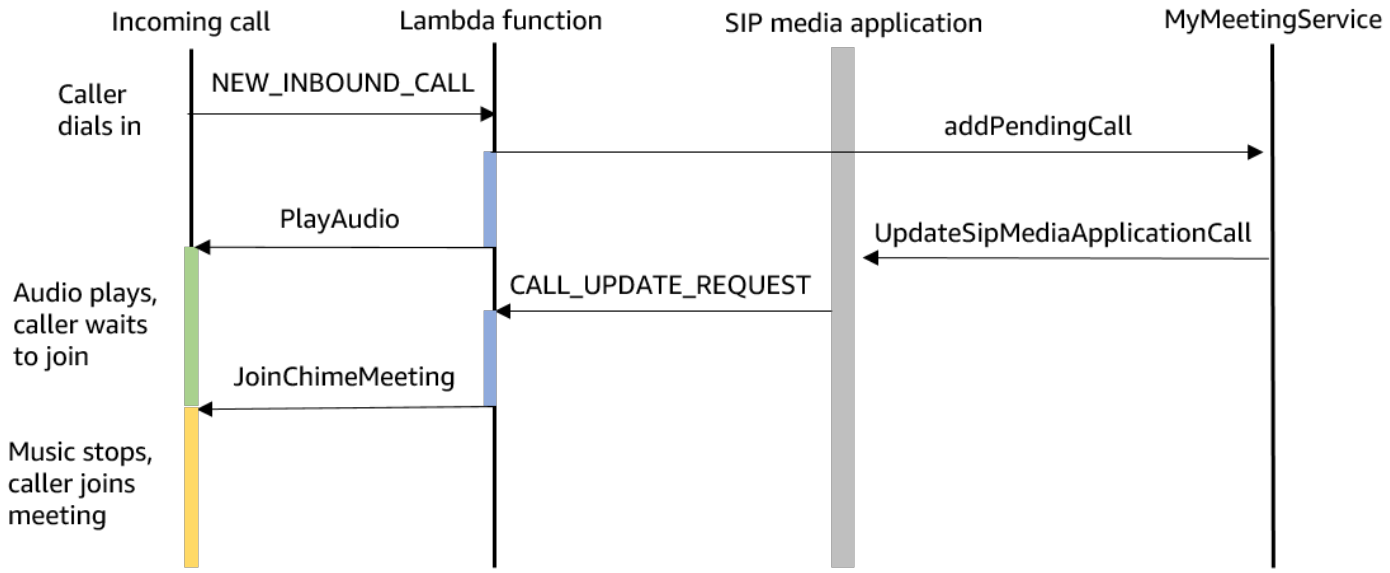
Workflow

You use the UpdateSipMediaApplicationCall API in variety of cases, such as adding participants to a meeting, muting and unmuting user, disconnecting them, and so on. The following use case describes a typical workflow.

A user calls and listens to music while Amazon Chime SDK sets up the meeting. Once setup completes, Amazon Chime SDK stops the audio and admits the caller into the meeting. Next, assume the use of a separate system, `MyMeetingService`, that manages meetings. Every incoming call should be put on hold. Chime notifies MyMeetingService about incoming calls, and MyMeetingService then creates an attendee for each call, and when the MyMeetingService is ready to start the meeting, it notifies the SIP media application and provides a token for joining the meeting.

To handle this case, the Lambda function has to implement the following logic.

- When a new incoming call arrives, the Lambda is invoked with a `NEW_INBOUND_CALL` event. The Lambda calls the `MyMeetingService` and passes the `transactionId` that identifies the current call, and returns the `PlayAudio` action.
- When the `MyMeetingService` is ready to add the caller to the meeting, the service calls the UpdateSipMediaApplicationCall API and passes the call's `transactionId` and `JoinToken` as part of its arguments. This API call triggers the Lambda function again, now with the `CALL_UPDATE_REQUESTED` event. The MyMeetingService passes the `JoinToken` to the Lambda function as part of the event, and the token is used to return the `JoinChimeMeeting` action to the SIP media application, which interrupts the `PlayAudio` action and connects the caller to the meeting.



> **Note**
>
> The UpdateSipMediaApplicationCall API returns HTTP 202 (Accepted). The SIP media application confirms that the call is in progress and can be updated, so it attempts to invoke the Lambda function. The invocation is performed asynchronously, so a successful response from the API doesn't guarantee that the Lambda function has started or completed.

The following example shows the request syntax.

```
{
    "SipMediaApplicationId": "string",
    "TransactionId": "string",
    "Arguments": {
        "string": "string"
    }
}
```

Request parameters

- `SipMediaApplicationId` – The ID of the SIP media application that handles the call.
- `TransactionId` – The ID of the call transaction. For inbound calls, the `TransactionId` can be obtained from the `NEW_INCOMING_CALL` event passed to the Lambda function

on its first invocation. For outbound calls, `TransactionId` is returned in the response of CreateSipMediaApplicationCall.

- **Arguments** – Custom arguments made available to the Lambda function as part of the `CallUpdateRequest` action data. Can contain 0 to 20 key-value pairs.

The following example shows a typical request.

```
aws chime update-sip-media-application-call --sip-media-application-id
 feb37a7e-2b66-49fb-b2dd-30f4780dc36d --transaction-id 1322a4e7-c106-4e70-aaaf-
a8fa4c77c0cb --arguments '{"JoinToken": "abc123"}'
```

Response syntax

```
{
  "SipMediaApplicationCall": {
  "TransactionId": "string"
  }
}
```

Response elements

- **TransactionId** – The ID of the call transaction, the same ID as the request.

The following example shows a `CALL_UPDATE_REQUESTED` invocation event.

```
{
  "SchemaVersion": "1.0",
  "Sequence": 2,
  "InvocationEventType": "CALL_UPDATE_REQUESTED",
  "ActionData": {
    "Type": "CallUpdateRequest",
    "Parameters": {
      "Arguments": {
        "string": "string"
      }
    }
  },
  "CallDetails": {
    ...
  }
```

```
}
```

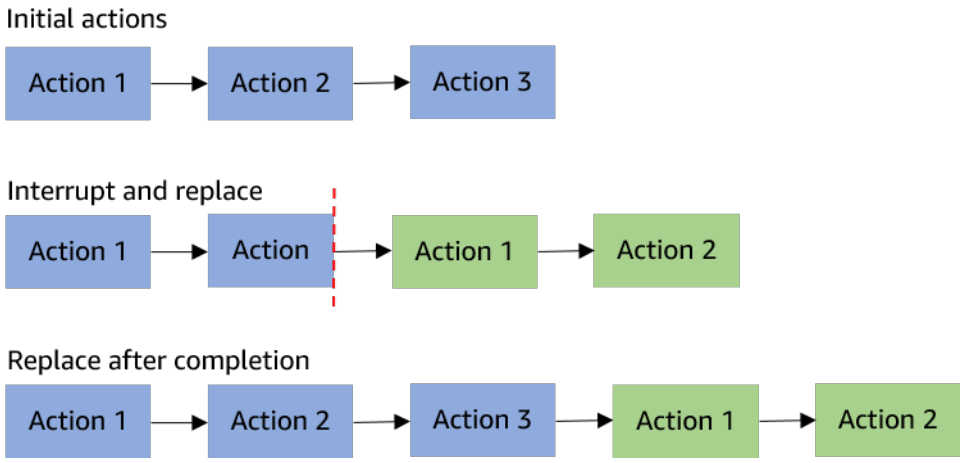
Event elements

- **SchemaVersion** – The version of the JSON schema (1.0)
- **Sequence** – The sequence number of the event in the call
- **InvocationEventType** – The type of Lambda invocation event, in this case, `CALL_UPDATE_REQUESTED`
- **ActionData** – The data associated with the `CallUpdateRequest` action.
  - **Type** – The type of action, in this case, `CallUpdateRequest`
  - **Parameters** – The parameters of the action
    - **Arguments** – The arguments passed as part of the `UpdateSipMediaApplicationCall` API request
- **CallDetails** – The information about the current call state

Understanding interruptible and non-interruptable actions

When a Lambda function returns a new list of actions while existing actions run, all actions that follow the in-progress action are replaced with the new actions. In some cases, the Lambda function interrupts in-progress actions in order to run new actions immediately.

The following diagram shows a typical example. Text below the digram explains the logic.



If Action 2 is interruptible, we stop it and run new Action 1 instead.

If Action 2 is not interruptible, it completes before the new Action 1 starts.

In both cases, Action 3 isn't run.

If something interrupts an action, the Lambda function is invoked with an `ACTION_INTERRUPTED` event. This event is used for informational purpose only. The SIP media application ignores all actions returned by this invocation.

Types of interruptible actions:

- `PlayAudio`
- `RecordAudio`
- `Pause`

Sample Lambda function

This example shows a typical Lambda function that plays an audio file, passes a join token, and updates the call.

```
const MMS = require('my-meeting-service');
const myMeetingServiceClient = new MMS.Client();

exports.handler = async (event) => {
    console.log('Request: ' + JSON.stringify(event));

    const playAudio = () => {
      return {
        Type: 'PlayAudio',
        Parameters: {
          ParticipantTag: 'LEG-A',
          AudioSource: {
            Type: 'S3',
            BucketName: 'chime-meetings-audio-files-bucket-name',
            Key: 'welcome.wav'
          }
        }
      }
    }

    const joinChimeMeeting = (joinToken) => {
      return {
        Type: 'JoinChimeMeeting',
        Parameters: {
```

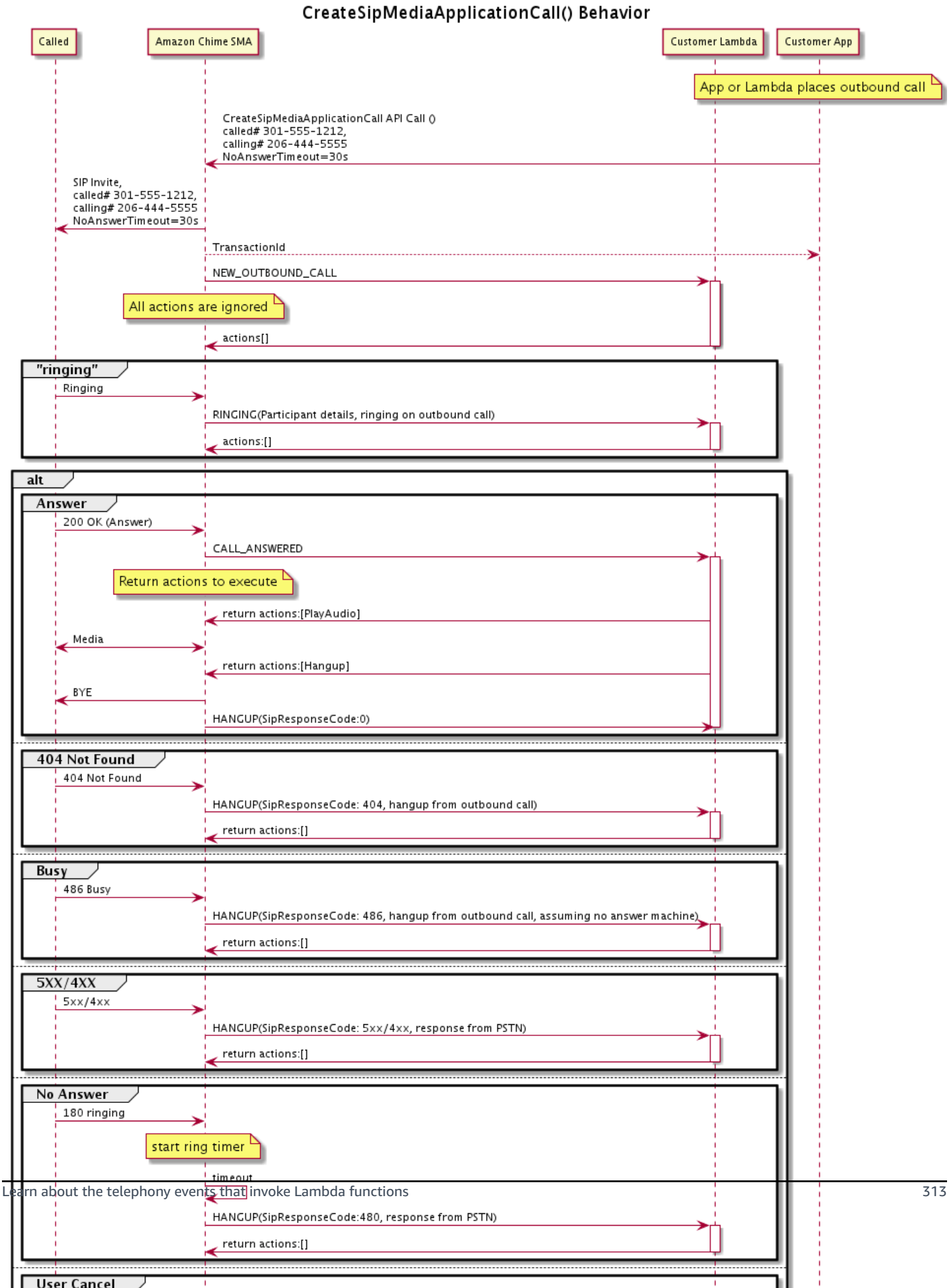
```
          JoinToken: joinToken
        }
      }
    }

    const response = (...actions) => {
      const r = {
        SchemaVersion: '1.0',
        Actions: actions
      };
      console.log('Response: ' + JSON.stringify(r));
      return r;
    };

    switch (event.InvocationEventType) {
      case 'NEW_INBOUND_CALL':
        myMeetingServiceClient.addPendingCall(event.CallDetails.TransactionId);

        return response(playAudio());
      case 'CALL_UPDATE_REQUESTED':
        const joinToken = event.ActionData.Parameters.Arguments['JoinToken']
        return response(joinChimeMeeting(joinToken));
      default:
        return response();
    }
}
```

Ending a call

You can use the CreateSipMediaApplicationCall API to end an outbound call. The API invokes the endpoint of a specified **SIP media application ID**. Customers can control the flow of the call by returning actions to the SIP media application.

In the event of a successful response, the API returns a 202 http status code along with the `transactionId`, which you can use with the UpdateSipMediaApplicationCall API to update an in-progress call.

The following diagram shows the invocations made to the AWS Lambda function endpoint for an outbound call.


CreateSipMediaApplicationCall() Behavior
Called
Amazon Chime SMA
Customer Lambda
Customer App
App or Lambda places outbound call
CreateSipMediaApplicationCall API Call ()
called# 301-555-1212,
calling# 206-444-5555
NoAnswerTimeout=30s
SIP Invite,
called# 301-555-1212,
calling# 206-444-5555
NoAnswerTimeout=30s
TransactionId
NEW_OUTBOUND_CALL
All actions are ignored
actions[]
"ringing"
Ringing
RINGING(Participant details, ringing on outbound call)
actions:[]
alt
Answer
200 OK (Answer)
CALL_ANSWERED
Return actions to execute
return actions:[PlayAudio]
Media
return actions:[Hangup]
BYE
HANGUP(SipResponseCode:0)
404 Not Found
404 Not Found
HANGUP(SipResponseCode: 404, hangup from outbound call)
return actions:[]
Busy
486 Busy
HANGUP(SipResponseCode: 486, hangup from outbound call, assuming no answer machine)
return actions:[]
5XX/4XX
5xx/4xx
HANGUP(SipResponseCode: 5xx/4xx, response from PSTN)
return actions:[]
No Answer
180 ringing
start ring timer
timeout
HANGUP(SipResponseCode:480, response from PSTN)
return actions:[]
User Cancel

The endpoint configured for the SIP media application is invoked for different statuses of the outbound call. When a customer ands a call, the Amazon Chime SDK invokes the endpoint with a HANGUP invocation event type.

This example shows a typical invocation event for a HANGUP.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 6,
    "InvocationEventType": "HANGUP",
    "ActionData": {
        "Type": "Hangup",
        "Parameters": {
            "CallId": "call-id-1",
            "ParticipantTag": "LEG-A"
        }
    },
    "CallDetails": {
        "TransactionId": "transaction-id",
        "AwsAccountId": "aws-account-id",
        "AwsRegion": "us-east-1",
        "SipRuleId": "sip-rule-id",
        "SipApplicationId": "sip-application-id",
        "Participants": [
            {
                "CallId": "call-id-1",
                "ParticipantTag": "LEG-A",
                "Direction": "Inbound",
                 "To": "+12065551212",
                "From": "+15105550101",
                "StartTimeInMilliseconds": "1597009588",
                "Status": "Disconnected"
            }
        ]
    }
}

// if LEG-B receives a hangup in a bridged call, such as a meeting ending
{
    "SchemaVersion": "1.0",
    "Sequence": 6,
    "InvocationEventType": "HANGUP",
    "ActionData": {
```

```
        "Type": "ReceiveDigits",
        "Parameters": {
            "CallId": "call-id-2",
            "ParticipantTag": "LEG-B"
        }
    },
    "CallDetails": {
        "TransactionId": "transaction-id",
        "AwsAccountId": "aws-account-id",
        "AwsRegion": "us-east-1",
        "SipRuleId": "sip-rule-id",
        "SipApplicationId": "sip-application-id",
        "Participants": [
            {
                "CallId": "call-id-1",
                "ParticipantTag": "Leg-A",
                 "To": "+12065551212",
                "From": "+15105550101",
                "Direction": "Inbound",
                "StartTimeInMilliseconds": "1597009588",
                "Status": "Connected"
            },
            {
                "CallId": "call-id-2",
                "ParticipantTag": "Leg-B",
                "To": "+17035550122",
                "From": "SMA",
                "Direction": "Outbound",
                "StartTimeInMilliseconds": "15010595",
                "Status": "Disconnected"
            }
        ]
    }
}
```

Understanding end-to-end calls

This use case provides example code for receiving a phone call from a PSTN caller, greeting the caller with an audio message, getting the meeting PIN from the caller, playing audio, and joining the caller to the meeting.

Invocation events and actions

The Audio Service passes invocation events to AWS Lambda functions as JSON objects. The objects include the invocation event type and any relevant metatdata. The AWS Lambda function also returns SIP media application actions as JSON objects, and those objects include an action type and any relevant metadata.

The following table lists the invocation events, and the possible `ActionData.Type`, when you receive an invocation event.

Invocation event	ActionData.Type
ACTION_SUCCESSFUL	CallAndBridge ReceiveDigits PlayAudio PlayAudioAndGetDigits JoinChimeMeeting ModifyChimeMeetingAttendees RecordMeeting
ACTION_FAILED	CallAndBridge PlayAudio PlayAudioAndGetDigits ModifyChimeMeetingAttendees RecordMeeting
HANGUP	HangUp
DIGITS_RECEIVED	ReceiveDigits

> **Note**
>
> To implement the following use case, you need at least one phone number in your Amazon Chime SDK inventory, a SIP media application managed object that uses a AWS Lambda function with an Amazon Resource Name (ARN), and a SIP rule that uses the phone number as its trigger.

When Amazon Chime SDK receives a call to the phone number specified in the rule, the PSTN Audio service invokes a AWS Lambda function with the `NEW_INBOUND_CALL` invocation event type.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 1,
    "InvocationEventType": "NEW_INBOUND_CALL",
    "CallDetails": {
        "TransactionId": "transaction-id",
        "AwsAccountId": "aws-account-id",
        "AwsRegion": "us-east-1",
        "SipRuleId": "sip-rule-id",
        "SipApplicationId": "sip-application-id",
        "Participants": [
            {
                "CallId": "call-id-1",
                "ParticipantTag": "LEG-A",
                "To": "+11234567890",
                "From": "+19876543210",
                "Direction": "Inbound",
                "StartTimeInMilliseconds": "159700958834234",
                "Status": "Connected"
            }
        ]
    }
}
```

You can program the AWS Lambda function to validate call details and store them for future use. For a `NEW_INBOUND_CALL` event, the AWS Lambda function responds with a set of actions that play a welcome prompt and ask for the meeting PIN.

Audio files have the following requirements:

- You must play audio files from an Amazon Simple Storage Service (S3) bucket. The S3 bucket must belong to the same AWS account as the SIP media application. In addition, you must give the `s3:GetObject` permission to the Amazon Chime SDK Voice Connector service principal —`voiceconnector.chime.amazonaws.com`. You can use the S3 console or the command-line interface (CLI) to do that.
- You must use PCM WAV files of no more than 50 MB in size. The Amazon Chime SDK recommends 8 KHz mono.
- The S3 metadata for each WAV file must contain `{'ContentType': 'audio/wav'}`.

```
{
    "SchemaVersion": "1.0",
    "Actions": [
        {
            "Type" : "PlayAudio",
            "Parameters" : {
                "CallId": "call-id-1",

                "AudioSource": {
                    "Type": "S3",
                    "BucketName": "chime-meetings-audio-files-bucket-name",
                    "Key": "welcome-to-meetings.wav"
                }
            }
        },
        {
            "Type": "PlayAudioAndGetDigits",
            "Parameters" : {
                "ParticipantTag": "LEG-A",

                "AudioSource": {
                    "Type": "S3",
                    "BucketName": "chime-meetings-audio-files-bucket-name",
                    "Key": "enter-meeting-pin.wav"
                },
                "FailureAudioSource": {
                    "Type": "S3",
                    "BucketName": "chime-meetings-audio-files-bucket-name",
                    "Key": "invalid-meeting-pin.wav"
                },
                "MinNumberOfDigits": 3,
                "MaxNumberOfDigits": 5,
```

```
                "TerminatorDigits": ["#"],
                "InBetweenDigitsDurationInMilliseconds": 5000,
                "Repeat": 3,
                "RepeatDurationInMilliseconds": 10000
            }
        }
    ]
}
```

The SIP media application runs these actions on call leg A. Assuming the `PlayAudioAndGetDigits` action receives the digits, the SIP media application invokes the AWS Lambda function with the `ACTION_SUCCESSFUL` event type.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 2,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type": "PlayAudioAndGetDigits",
        "Parameters" : {
            "ParticipantTag": "LEG-A",
            "AudioSource": {
                "Type": "S3",
                "BucketName": "chime-meetings-audio-files-bucket-name",
                "Key": "enter-meeting-pin.wav"
            },
            "FailureAudioSource": {
                "Type": "S3",
                "BucketName": "chime-meetings-audio-files-bucket-name",
                "Key": "invalid-meeting-pin.wav"
            },
            "MinNumberOfDigits": 3,
            "MaxNumberOfDigits": 5,
            "TerminatorDigits": ["#"],
            "InBetweenDigitsDurationInMilliseconds": 5000,
            "Repeat": 3,
            "RepeatDurationInMilliseconds": 10000
        },
        "ReceivedDigits": "12345" // meeting PIN
    },
    "CallDetails": {
        ... // same as in previous event
    }
```

```
}
}
```

You can program an AWS Lambda function to identify the caller based on the `CallDetails` data. You can also validate the meeting PIN received earlier. Assuming a correct PIN, you then use the CreateMeeting and CreateAttendee APIs to create the Amazon Chime SDK meeting and generate the join token used by the meeting attendee. The AWS Lambda function responds with the action to join the Amazon Chime SDK meeting.

```
{
    "SchemaVersion": "1.0",
    "Actions": [
        {
            "Type": "JoinChimeMeeting",
            "Parameters": {
                "JoinToken": "meeting-attendee-join-token"
            }
        }
    ]
}
```

Assuming the `JoinToken` is valid, the SIP media application joins the Amazon Chime SDK meeting and invokes a AWS Lambda function with the `ACTION_SUCCESSFUL` event, where `CallDetails` contains the data from the SIP media application and the Chime Media service (`LEG-B`)

```
{
    "SchemaVersion": "1.0",
    "Sequence": 3,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type" : "JoinChimeMeeting",
        "Parameters" : {
            "JoinToken": "meeting-attendee-join-token"
        }
    },
    "CallDetails": {
        "TransactionId": "transaction-id",
        "AwsAccountId": "aws-account-id",
        "AwsRegion": "us-east-1",
        "SipRuleId": "sip-rule-id",
        "SipApplicationId": "sip-application-id",
        "Participants": [
```

```
            {
                "CallId": "call-id-1",
                "ParticipantTag": "LEG-A",
                "To": "+11234567890",
                "From": "+19876543210",
                "Direction": "Inbound",
                "StartTimeInMilliseconds": "159700958834234",
                "Status": "Connected"
            },
            {
                "CallId": "call-id-2",
                "ParticipantTag": "LEG-B",
                "To": "SMA",
                "From": "+17035550122",
                "Direction": "Outbound",
                "StartTimeInMilliseconds": "159700958834234",
                "Status": "Connected"
            }
        ]
    }
}
```

If you want to stop running actions on the call or call leg at this point, you can respond with an empty set of actions.

```
{
    "SchemaVersion": "1.0"
    "Actions": []
}
```

After the caller hangs up, the SIP media application invokes the AWS Lambda function with the HANGUP event.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 4,
    "InvocationEventType": "HANGUP",
    "ActionData": {
        "Type": "Hangup",
        "Parameters": {
            "CallId": "call-id-1",
            "ParticipantTag": "LEG-A"
```

```
        }
    },
    "CallDetails": {
        "TransactionId": "transaction-id",
        "AwsAccountId": "aws-account-id",
        "AwsRegion": "us-east-1",
        "SipRuleId": "sip-rule-id",
        "SipApplicationId": "sip-application-id",
        "Participants": [
            {
                "CallId": "call-id-1",
                "ParticipantTag": "LEG-A",
                "To": "+11234567890",
                "From": "+19876543210",
                "Direction": "Inbound",
                "StartTimeInMilliseconds": "159700958834234",
                "Status": "Disconnected"
            },
            {
                "CallId": "call-id-2",
                "ParticipantTag": "LEG-B",
                "To": "SMA",
                "From": "+17035550122",
                "Direction": "Outbound",
                "StartTimeInMilliseconds": "159700958834234",
                "Status": "Disconnected"
            }
        ]
    }
}
```

If you respond to a Hangup event with an action, the SIP media application ignores the action if no other `Participants` show a `Status` of `Connected`.

Responding to invocations with action lists

You can respond to an AWS Lambda invocation event with a list of actions to run on the individual participants in a call. You can respond with a maximum of 10 actions for each AWS Lambda invocation, and you can invoke an AWS Lambda function 1,000 times per call.

By default, SIP media applications time out if a Lambda function doesn't respond after 20 seconds.

The following example shows the general response structure.

```
{
    "SchemaVersion": "1.0",
    "Actions": [
        {
            "Type": "PlayAudio",
            "Parameters": {
                "ParticipantTag": "LEG-A",
                "AudioSource": {
                    "Type": "S3",
                    "BucketName": "bucket-name",
                    "Key": "audio-file.wav"
                }
            }
        },
        {
            "Type": "RecordAudio",
            "Parameters": {
                "DurationInSeconds": "10",
                "RecordingTerminators": ["#"],
                "RecordingDestination": {
                    "Type": "S3",
                    "BucketName": "bucket-name"
                }
            }
        }
    ]
}
```

When the AWS Lambda function returns the list of actions to the SIP media application, the following operations occur:

1. The application finishes running the current action on a call.
2. The application then replaces the old action set with a new set of actions received from the latest invocation event.

If the SIP media application receives a `NULL` action set, it keeps the existing actions.

Supported actions for the PSTN Audio service

You can specify different types of signaling and media actions in a response from an AWS Lambda function. Each action has different properties. The following topics provide example code and explain how to use the actions.

Contents

Using TransactionAttributes

You use the `TransactionAttributes` data structure to store application-specific information, such as call states or meeting IDs, and then pass that data to AWS Lambda invocations. This structure removes the need for storing data in external databases such as Amazon DynamoDB.

`TransactionAttributes` are JSON Objects that contain key/value pairs. The objects can contain a maximum of 100 key/value pairs, and the objects have a maximum payload size of 20 KB. The data in a `TransactionAttributes` structure persists for the life of a transaction.

When an AWS Lambda function passes `TransactionAttributes` to a SIP media application, the application updates any stored attributes. If you pass a `TransactionAttributes` object with

an existing key set, you update the stored values. If you pass a different key set, you replace the existing values with the values from that different key set. Passing an empty map (`{}`) erases any stored values.

Topics

- Setting TransactionAttributes
- Updating TransactionAttributes
- Clearing TransactionAttributes
- Handling ACTION_SUCCESSFUL events
- Invalid inputs

Setting TransactionAttributes

The following example shows how to set `TransactionAttributes` alongside a PlayAudio action and pass the attributes from an AWS Lambda function to a SIP media application.

```
{
    "SchemaVersion": "1.0",
    "Actions": [
        {
            "Type": "PlayAudio",
            "Parameters": {
                "ParticipantTag": "LEG-A",
                "AudioSource": {
                    "Type": "S3",
                    "BucketName": "mtg1-sipmedia-app-iad",
                    "Key": "Welcome3.wav"
                }
            }
        }
    ],
    "TransactionAttributes": {
        "key1": "value1",
        "key2": "value2"
    }
}
```

Updating TransactionAttributes

To modify stored `TransactionAttributes`, update the contents of the JSON object with new values. In the following example, the keys `NewKey1` and `NewKey2` are added to the `TransactionAttributes`. These keys are paired with the values `NewValue1` and `NewValue2`, respectively.

```
{
    "SchemaVersion": "1.0",
    "Actions": [
        {
            "Type": "PlayAudio",
            "Parameters": {
                "ParticipantTag": "LEG-A",
                "AudioSource": {
                    "Type": "S3",
                    "BucketName": "mtg1-sipmedia-app-iad",
                    "Key": "Welcome3.wav"
                }
            }
        }
    ],
    "TransactionAttributes": {
        "NewKey1": "NewValue1",
        "NewKey2": "NewValue2"
    }
}
```

If, in the previous example, you passed `NewValue1` to `key1`, the existing value of `key1` would be replaced with `NewValue1`. However, passing a value to `NewKey1` creates a new key/value pair.

Clearing TransactionAttributes

To clear the contents of the `TransactionAttributes` object, pass the `TransactionAttributes` field with an empty JSON Object:

```
{
    "SchemaVersion": "1.0",
    "Actions": [
        {
            "Type": "PlayAudio",
            "Parameters": {
```

```
                "ParticipantTag": "LEG-A",
                "AudioSource": {
                    "Type": "S3",
                    "BucketName": "mtg1-sipmedia-app-iad",
                    "Key": "Welcome3.wav"
                }
            }
        }
    ],
    "TransactionAttributes": {
    }
}
```

> **Note**
>
> You can't clear data from a `TransactionAttributes` structure by setting its value to `null`. Also, omitting the `TransactionAttribute` structure doesn't clear its data. Always pass an empty JSON object with `TransactionAttributes` to clear data from the object.

Handling ACTION_SUCCESSFUL events

The following example shows how a successful PlayAudio sends the stored `TransactionAttributes` as part of the `CallDetails` .

```
{
    "SchemaVersion": "1.0",
    "Sequence": 2,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type": "PlayAudio",
        "Parameters": {
            "AudioSource": {
                "Type": "S3",
                "BucketName": "mtg1-sipmedia-app-iad",
                "Key": "Welcome3.wav"
            },
            "Repeat": 1,
            "ParticipantTag": "LEG-A"
        }
    },
    "CallDetails": {
```

```
        "TransactionId": "mtg1-tx-id",
        "TransactionAttributes": {
            "key1": "value1",
            "key2": "value2"
        },
        "AwsAccountId": "166971021612",
        "AwsRegion": "us-east-1",
        "SipRuleId": "aafbd402-b7a2-4992-92f8-496b4563c492",
        "SipMediaApplicationId": "e88f4e49-dd21-4a3f-b538-bc84eae11505",
        "Participants": [
            {
                "CallId": "bbff30c5-866a-41b5-8d0a-5d23d5e19f3e",
                "ParticipantTag": "LEG-A",
                "To": "+14345550101",
                "From": "+14255550199",
                "Direction": "Inbound",
                "StartTimeInMilliseconds": "1644539405907",
                "Status": "Connected"
            }
        ]
    }
}
```

Invalid inputs

The following example shows an invalid input. In this case, the JSON object passes too many items to a SIP media application.

```
{
    "SchemaVersion": "1.0",
    "Actions": [
        {
            "Type": "PlayAudio",
            "Parameters": {
                "ParticipantTag": "LEG-A",
                "AudioSource": {
                    "Type": "S3",
                    "BucketName": "mtg1-sipmedia-app-iad",
                    "Key": "Welcome3.wav"
                }
            }
        }
    ],
```

```
    "TransactionAttributes": {
        "key1": "value1",
        "key2": "value2",
        "key3": "value3",
        "key4": "value4",
        "key5": "value5",
        "key6": "value6",
        "key7": "value7",
        "key8": "value8",
        "key9": "value9",
        "key10": "value10",
        "key11": "value11"
    }
}
```

The following example shows the response to the previously given input. This output is passed from a SIP media application back to the AWS Lambda function that invoked the application.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 2,
    "InvocationEventType": "INVALID_LAMBDA_RESPONSE",
    "CallDetails": {
        "TransactionId": "mtg1-tx-id",
        "AwsAccountId": "166971021612",
        "AwsRegion": "us-east-1",
        "SipRuleId": "aafbd402-b7a2-4992-92f8-496b4563c492",
        "SipMediaApplicationId": "e88f4e49-dd21-4a3f-b538-bc84eae11505",
        "Participants": [
            {
                "CallId": "72cbec69-f098-45d8-9ad6-e26cb9af663a",
                "ParticipantTag": "LEG-A",
                "To": "+14345550101",
                "From": "+14255550199",
                "Direction": "Inbound",
                "StartTimeInMilliseconds": "1644540839987"
            }
        ]
    },
    "ErrorType": "TransactionAttributesInvalidMapSize",
    "ErrorMessage": "Transaction Attributes has too many mappings. Maximum number of
 mappings is 10"
}
```

Using call recording

The call recording actions for SIP media applications enable you to build call recording and post-call transcription solutions for a variety of uses. For example, you can record customer-care calls and use them for training.

You use the call recording actions in concert with your SIP media applications. You can also use the actions on-demand or in response to a SIP event.

- To start on-demand recording of a call in your SIP media application, you use the UpdateSipMediaApplication API to invoke your application and return the StartCallRecording action.
- To start call recording in response to a SIP event, you return the `StartCallRecording` action in your application.

You can pause and resume an in-progress recording. To pause, use the PauseCallRecording action. To resume, use the `ResumeCallRecording` action. Each time you pause or resume a recording, the action captures a tone that indicates the pause or resumption. When you pause, the action records silence, which Amazon Chime SDK uses to track the length of the pause and include the pauses in your bill. You can pause and resume recording as often as needed.

To stop call recording, you return the StopCallRecording action. However, call recordings automatically stop when the call stops, and in that case you don't need to explicitly return the `StopCallRecording` action. You can only start and stop recording once for an individual call leg.

Amazon Chime SDK delivers call recordings to an Amazon S3 bucket that you select. The bucket must belong to your AWS account. Once a call stops, the SIP media application delivers the recording to the folder specified in the `Destination` parameter of the StartCallRecording action. The Amazon Chime SDK records calls in an open WAV format. Calls that record incoming and outgoing tracks use stereo mode, with the incoming track in the left channel and the outgoing track on the right channel. If you record only the incoming or outgoing track, the system uses mono mode.

> **Note**
>
> Recordings made using this feature may be subject to laws or regulations regarding the recording of electronic communications. It is your and your end users' responsibility to comply with all applicable laws regarding the recording, including properly notifying all

participants in a recorded session or communication that the session or communication is being recorded, and obtaining their consent.

Billing for call recording

Amazon Chime SDK bills you per minute for the time that call recording is enabled for a call leg, and that time includes all pauses. You are billed for the call recording usage once the call recording is delivered to your Amazon S3 bucket.

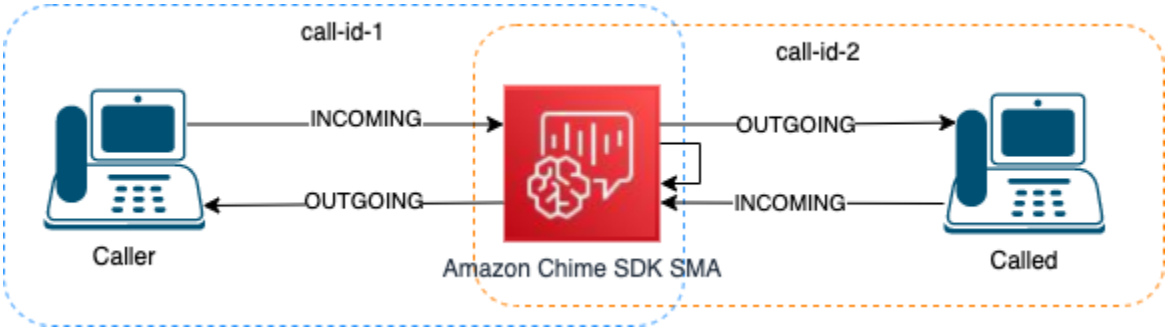
Recording audio tracks

You can record just the incoming or outgoing tracks of call, or both tracks of a call.

This image shows a typical one-legged, or non-bridged, incoming call.

The call only has one leg with a `callID` of **call-id-1**. The `INCOMING` audio track is the audio from the caller to the SIP media application. The `OUTGOING` audio track is the audio from the SIP media application to the caller. Your SIP media application specifies the `CallId` of the call that you want record. To record the participant who placed the call, you specify `INCOMING`. To record the participant who responds to a call, you specify `OUTGOING`. To record both participants, specify `BOTH`.

This image shows a typical bridged call with two participants.



In this example, the call has two call legs, **call-id-1** and **call-id-2**, and **call-id-1** is bridged to **call-id-2**. This creates four audio tracks, the incoming and outgoing audio streams for both call IDs. You

can specify which of the call IDs and audio tracks to record. For example, if you want to record the audio track from the called participant, you record the `INCOMING` audio track by specifying **call-id-2** as the `CallId` and `INCOMING` as the track.

If you want to record everything that the caller hears, you record the `OUTGOING` audio track by specifying **call-id-1** as the `CallId` and `OUTGOING` as the track. If you want to record all of the audio that the `Caller` said and heard, you record `BOTH` audio tracks by specifying `call-id-1` as the `CallId` and `BOTH` as the track.

Sample use cases

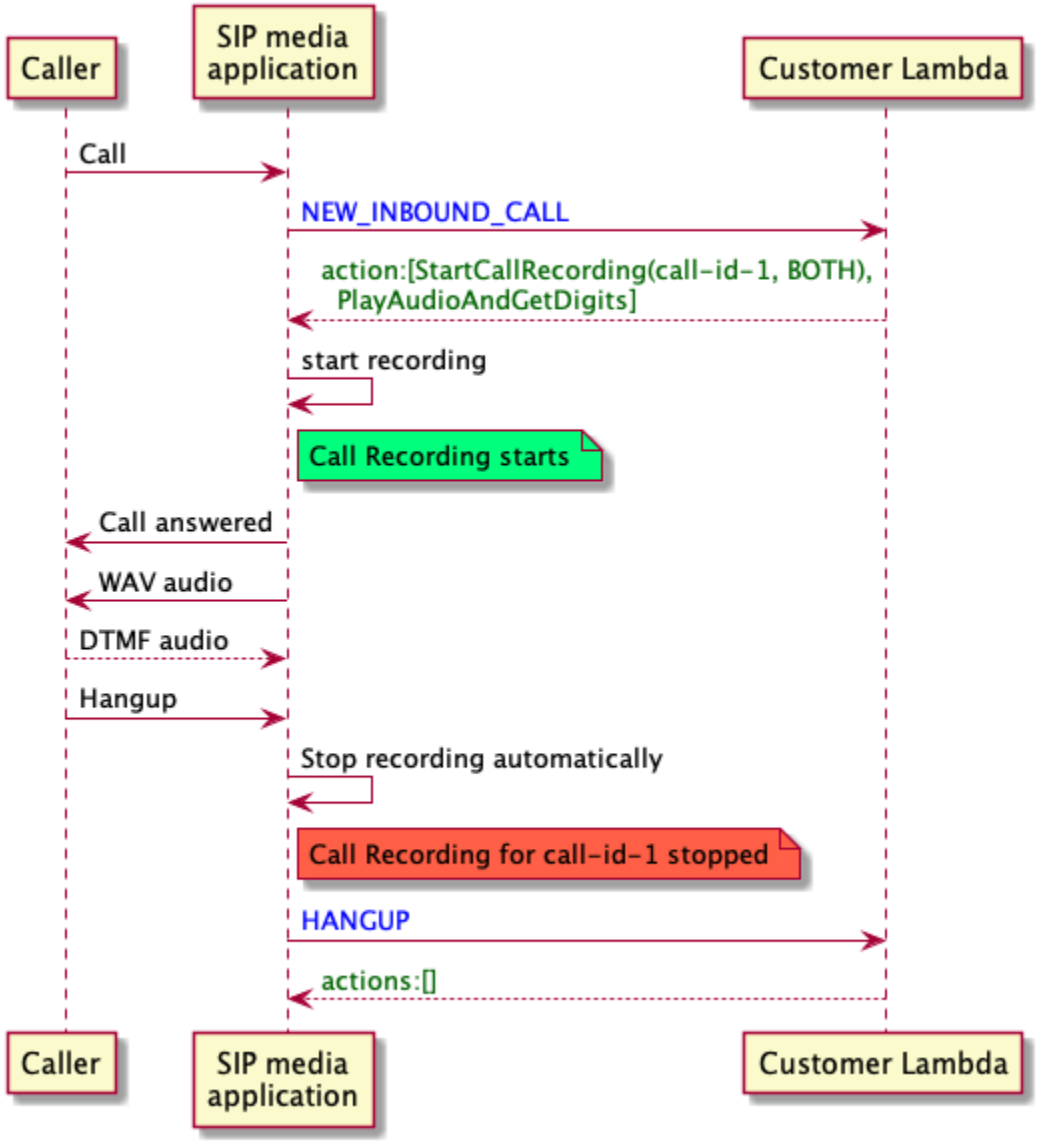
SIP media applications provide call recording actions as building blocks. They give you the flexibility to build call-recording solutions for your business use cases. The following cases illustrate some common usage scenarios.

Topics

Case 1: Recording a one-legged call that involves SIP actions

You can record a caller and any audio generated by SIP media application actions, such as the PlayAudio and PlayAudioAndGetDigits actions. During recording, if a caller presses a digit, the recording captures the tone of that digit. This example uses the `PlayAudioAndGetDigits` action, but the interactive voice response (IVR) can be a complex series of SIP media application actions.

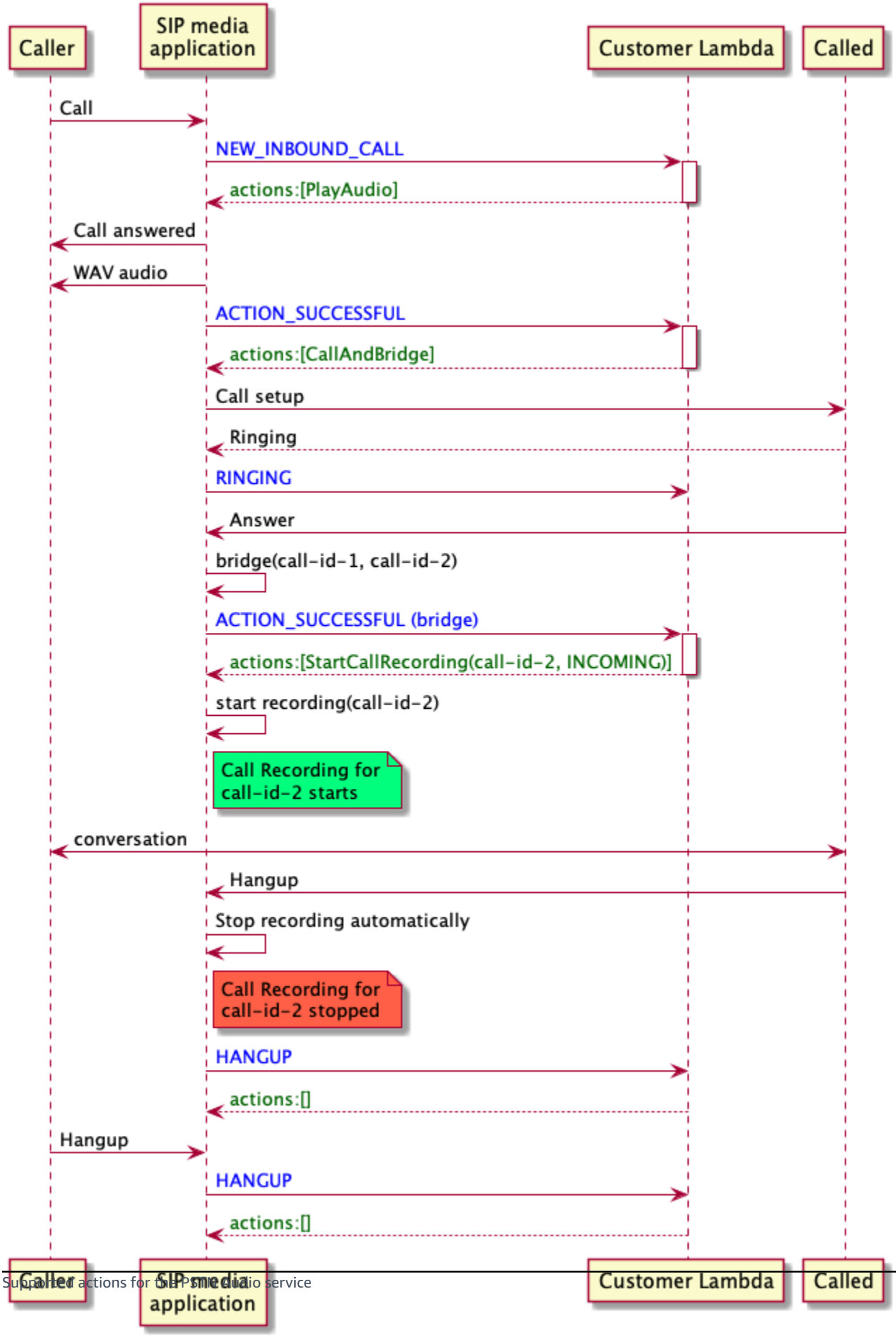
In this example, the SIP media application records both audio tracks between the caller and the SIP media application itself. The recording starts when the call is established, and it stops when the caller hangs up. Billing starts when the call is established, and it stops when the caller hangs up.



Case 2: Selectively recording audio in a bridged call

You can selectively record the audio track of a single call participant. You can use this feature to selectively enable call recording only for a specific participant.

In this example, the SIP media application records the incoming audio tracks between the called party and the SIP media application itself by specifying **call-id-2** as the `CallId` and `INCOMING` as the track. The call recording starts when the caller is bridged to the called party, and that's also when billing starts. The recording stops when the called party hangs up, and that's also when billing ends. This recording only has the called party's audio track.

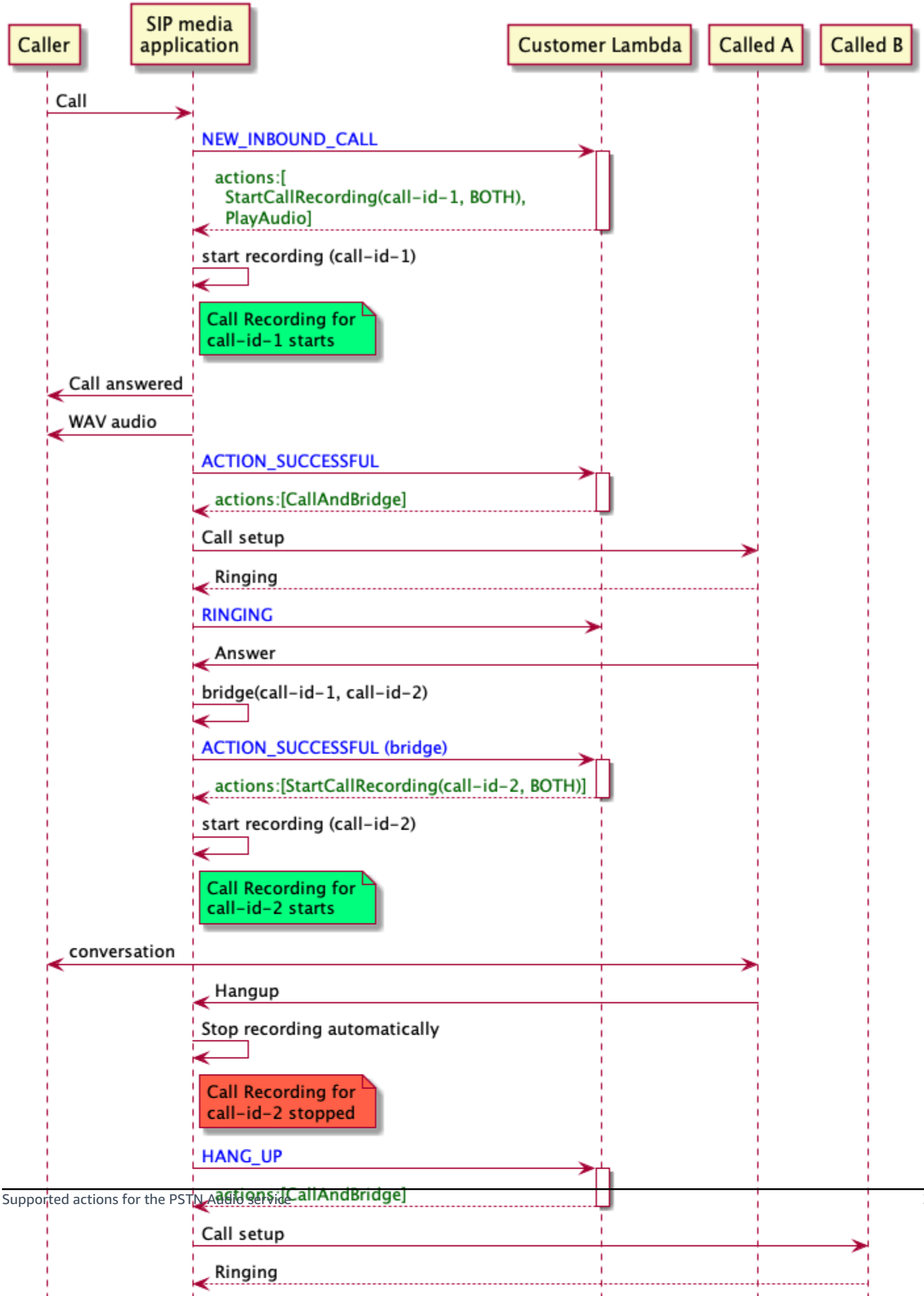

Caller
SIP media application
Customer Lambda
Called
Call
NEW_INBOUND_CALL
actions:[PlayAudio]
Call answered
WAV audio
ACTION_SUCCESSFUL
actions:[CallAndBridge]
Call setup
Ringing
RINGING
Answer
bridge(call-id-1, call-id-2)
ACTION_SUCCESSFUL (bridge)
actions:[StartCallRecording(call-id-2, INCOMING)]
start recording(call-id-2)
Call Recording for call-id-2 starts
conversation
Hangup
Stop recording automatically
Call Recording for call-id-2 stopped
HANGUP
actions:[]
Hangup
HANGUP
actions:[]
Caller
SIP media application
Customer Lambda
Called

Case 3: Recording multiple call legs

You can record multiple call legs. For example, say you bridge a call to a participant. When that participant hangs up, the phone call is bridged to another participant. You can enable call recording for all three call legs.

This example shows three separate recording files. The recording for the first call leg captures the conversation between the caller, your application, and the two participants that were bridged into the call. The recording for the second call leg captures the conversation between the caller and the first participant. The recording for the third call leg captures the conversation between the caller and the second participant.

This case creates three call legs, and billing applies to the start and end of each call leg. Put another way, the system delivers three recordings to your S3 bucket, and you're billed for each.

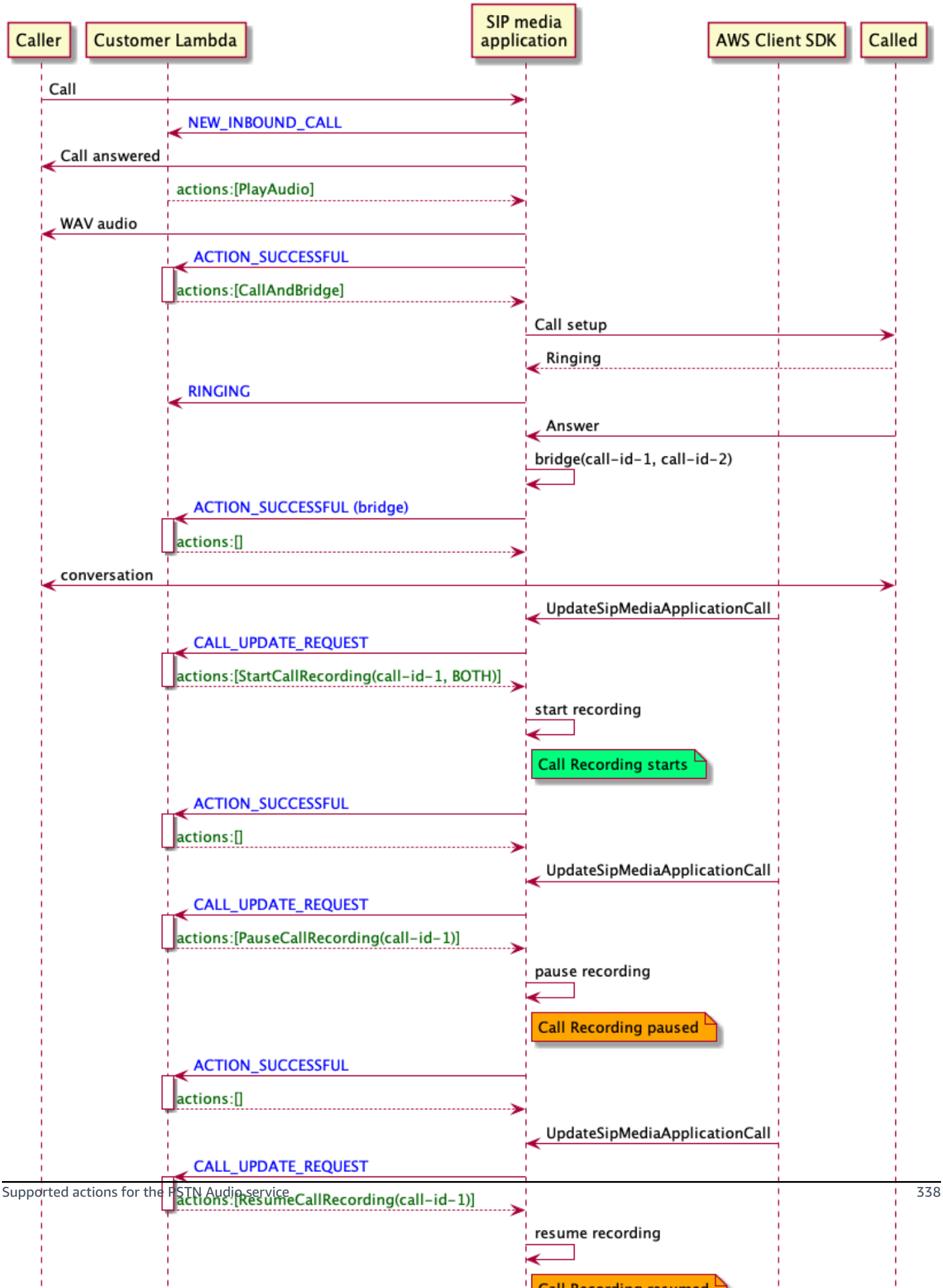

Caller
SIP media application
Customer Lambda
Called A
Called B
Call
NEW_INBOUND_CALL
actions:[
StartCallRecording(call-id-1, BOTH),
PlayAudio]
start recording (call-id-1)
Call Recording for call-id-1 starts
Call answered
WAV audio
ACTION_SUCCESSFUL
actions:[CallAndBridge]
Call setup
Ringing
RINGING
Answer
bridge(call-id-1, call-id-2)
ACTION_SUCCESSFUL (bridge)
actions:[StartCallRecording(call-id-2, BOTH)]
start recording (call-id-2)
Call Recording for call-id-2 starts
conversation
Hangup
Stop recording automatically
Call Recording for call-id-2 stopped
HANG_UP
actions:[CallAndBridge]
Call setup
Ringing

Case 4: On-demand recording with pause and resume

You can start, stop, pause, and resume call recording on-demand using the UpdateSipMediaApplicationCall API. You can build a client application that calls the `UpdateSipMediaApplicationCall` API and invokes your SIP media application to return call recording actions.

Your end users use the client application to control the call recording. For example, in a call center, an agent would use a desktop client application to trigger call recording actions on-demand. In the call center example, the agent might ask the caller's permission to record the phone call, and they can click in the client application to start recording once the caller agrees. In another example, the caller might need to provide information such as a social security number (SSN). However, the call center policy requires that the agent should not record information such as a customer's SSN. The agent can click the application to pause the recording while the customer provides the information, then click again to resume the recording. Once the agent handles the caller's request, the agent clicks the application to stop recording and hangs up the call.

In this use case, the SIP media application records the audio tracks between the caller and SIP media application. Since the `call-id-1` leg is bridged to the `call-id-2` leg, the system records the audio on both legs, caller and called. The recording and the billing start when the `UpdateSipMediaApplicationCall` API invokes the `StartCallRecording` action. The recording and the billing stop when the `UpdateSipMediaApplicationCall` API invokes the `StopCallRecording` action. As a reminder, pausing the recording does not change its duration, and you're billed for all pauses.


Caller
Customer Lambda
SIP media application
AWS Client SDK
Called
Call
NEW_INBOUND_CALL
Call answered
actions:[PlayAudio]
WAV audio
ACTION_SUCCESSFUL
actions:[CallAndBridge]
Call setup
Ringing
RINGING
Answer
bridge(call-id-1, call-id-2)
ACTION_SUCCESSFUL (bridge)
actions:[]
conversation
UpdateSipMediaApplicationCall
CALL_UPDATE_REQUEST
actions:[StartCallRecording(call-id-1, BOTH)]
start recording
Call Recording starts
ACTION_SUCCESSFUL
actions:[]
UpdateSipMediaApplicationCall
CALL_UPDATE_REQUEST
actions:[PauseCallRecording(call-id-1)]
pause recording
Call Recording paused
ACTION_SUCCESSFUL
actions:[]
UpdateSipMediaApplicationCall
CALL_UPDATE_REQUEST
actions:[ResumeCallRecording(call-id-1)]
resume recording

Call recording actions for SIP media applications

You can specify different call recording actions in a response from the AWS Lambda function of your SIP media application. The following topics provide example code and explain how to use the actions.

Topics

StartCallRecording

The `StartCallRecording` action starts the recording of a call leg. You start call recording in your SIP media applications, either on-demand or in response to a SIP event.

- To start on-demand recording of a call, you use the `UpdateSipMediaApplication` API to invoke your application and return the `StartCallRecording` action.
- To start call recording in response to a SIP event, you return the `StartCallRecording` action in your application.

You specify whether you want to record the audio track for the incoming leg, the outgoing leg, or both. The following sections explain how to use the `StartCallRecording` action.

> **Note**
>
> Recordings made using this feature may be subject to laws or regulations regarding the recording of electronic communications. It is your and your end users' responsibility to comply with all applicable laws regarding the recording, including properly notifying all participants in a recorded session or communication that the session or communication is being recorded, and obtaining their consent.

Topics

- Specifying a recording destination
- Granting Amazon S3 bucket permissions
- Action successful response
- Action error response

Requesting a StartCallRecording action

The following example shows how to request the `StartCallRecording` action for `BOTH` tracks.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Type": "StartCallRecording",
            "Parameters":
            {
                "CallId": "call-id-1",
                "Track": "BOTH",
                "Destination":
                {
                    "Type": "S3",
                    "Location": "valid-bucket-name-and-optional-prefix"
                }
            }
        }
    ]
}
```

CallId

Description – `CallId` of participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – Yes

Default value – None

Track

Description – Audio `Track` of the call recording.

Allowed values – `BOTH`, `INCOMING`, or `OUTGOING`

Required – Yes

Default value – None

Destination.Type

Description – Type of destination. Only Amazon S3 is allowed.

Allowed values – Amazon S3

Required – Yes

Default value – None

Destination.Location

Description – A valid Amazon S3 bucket and an optional Amazon S3 key prefix. The bucket must have permissions to the Amazon Chime SDK Voice Connector service principal, voiceconnector.chime.amazonaws.com.

Allowed values – A valid Amazon S3 path for which Amazon Chime SDK has permisions to the `s3:PutObject` and `s3:PutObjectAcl` actions.

Required – Yes

Default value – None

Specifying a recording destination

Amazon Chime SDK delivers call recordings to your Amazon S3 bucket. The bucket must belong to your AWS account. You specify the bucket's location in the `Destination` parameter of the `StartCallRecording` action. The `Type` field in the the `Destination` parameter must be `S3`. The `Location` field consists of your Amazon S3 bucket, plus an optional object-key prefix in which the call recording is delivered.

The SIP media application uses the specified `Location`, the call leg's date and time, the transaction ID, and the call ID to format the Amazon S3 object key. The `StartCallRecording` action response returns the full Amazon S3 object key.

When you only provide the Amazon S3 bucket in the `Location` field, the SIP media application appends a default prefix, `Amazon-Chime-SMA-Call-Recordings`, to the Amazon S3 path.

The SIP media application also appends the year, month, and day of the call's start time to help organize the recordings. The following example shows the general format of an Amazon S3 path with the default prefix. This example uses `myRecordingBucket` as the `Location` value.

```
myRecordingBucket/Amazon-Chime-SMA-Call-Recordings/2019/03/01/2019-03-01-17-10-00-
010_c4640e3b-1478-40fb-8e38-6f6213adf70b_7ab7748e-b47d-4620-ae2c-152617d3333c.wav
```

The following example shows the data represented in the call recording Amazon S3 path.

```
s3Bucket/Amazon-Chime-SMA-Call-Recordings/year/month/date/year-month-date-hour-minute-
second-millisecond_transactionId_callId.wav
```

When you provide the Amazon S3 bucket and object key prefix in the `Location` field, the SIP media application uses your object-key prefix in the destination Amazon S3 path instead of the default prefix. The following example shows the general format of a call recording Amazon S3 path with your prefix. For example, you can specify myRecordingBucket/technicalSupport/english as the `Location`.

```
myRecordingBucket/technicalSupport/english/2019/03/01/2019-03-01-17-10-00-
010_c4640e3b1478-40fb-8e38-6f6213adf70b_7ab7748e-b47d-4620-ae2c-152617d3333c.wav
```

The following example shows the data in the Amazon S3 path.

```
s3Bucket/yourObjectKeyPrefix/year/month/date/year-month-date-hour-minute-second-
millisecond_transactionId_callId.wav
```

The recording sent to your Amazon S3 bucket contains additional Amazon S3 object metadata about the call leg. The following table lists the supported Amazon S3 object metadata.

Name	Description
transaction-id	Transaction ID of the phone call
call-id	CallId of participant in the CallDetails of the AWS Lambda function invocation
recording-duration	Call recording duration in seconds

Name	Description
recording-audio-file-format	Call recording audio file format represented as internet media type

Granting Amazon S3 bucket permissions

Your destination Amazon S3 bucket must belong to the same AWS account as your application. In addition, the action must give `s3:PutObject` and `s3:PutObjectAcl` permission to the Amazon Chime SDK Voice Connector service principal, `voiceconnector.chime.amazonaws.com`. The following example grants the appropriate permission.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SIP media applicationRead",
            "Effect": "Allow",
            "Principal": {
                "Service": "voiceconnector.chime.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::bucket-name/*",
     "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "aws-account-id"
                }
            }
        }
    ]
}
```

The PSTN Audio Service reads and writes to your S3 bucket on behalf of your Sip Media Application. To avoid the confused deputy problem, you can restrict S3 bucket permissions to a single SIP media application.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SIP media applicationRead",
            "Effect": "Allow",
            "Principal": {
                "Service": "voiceconnector.chime.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::bucket-name/*",
     "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "aws-account-id",
                    "aws:SourceArn": "arn:aws:chime:region:aws-account-id:sma/sip-
media-application-id"
                }
            }
        }
    ]
}
```

Action successful response

When the call recording is successfully started on a call leg, the SIP media application invokes an AWS Lambda function with the `ACTION_SUCCESSFUL` event type. The location of call recording is returned in the response.

```
{
    "SchemaVersion": "1.0",
    "Sequence": INTEGER,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type" : "StartCallRecording",
        "Parameters": {
            "CallId": "call-id-1",
            "Track": "BOTH",
            "Destination": {
                "Type": "S3",
                "Location": "valid-bucket-name"
```

```
            }
        }
        "CallRecordingDestination": {
            "Type": "S3",
            "Location": "call-recording-bucket-and-key"
        }
    }
    "CallDetails": {
        ...
    }
}
```

Action error response

For validation errors, the SIP media application calls the AWS Lambda function with the appropriate error message. The following table lists the error messages.

Error	Message	Reason
`InvalidActionParameter`	`CallId` parameter for action is invalid	Any parameter is invalid.
`SystemException`	System error while running an action.	Another type of system error occurred while running an action.

When the action fails to record the media on a call leg, the SIP media application invokes an AWS Lambda function with the `ActionFailed` event type.

The following example shows a typical error response.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 5,
    "InvocationEventType": "ACTION_FAILED",
    "ActionData": {
        "Type" : "StartCallRecording",
        "Parameters": {
            "CallId": "call-id-1",
            "Track": "BOTH",
```

```
            "Destination": {
                "Type": "S3",
                "Location": "valid-bucket-name"
            }
        }
        "Error": "NoAccessToDestination: Error while accessing destination"
    }
    "CallDetails": {
        ...
    }
}
```

See a working example on GitHub: https://github.com/aws-samples/amazon-chime-sma-on-demand-recording

StopCallRecording

The `StopCallRecording` action stops the recording of a call leg. Recording stops automatically when a call ends, and your application doesn't need to explicitly return the `StopCallRecording` action. Once recording for a call leg stops, it can't start again, and the recording is delivered to the destination specified in the `StartCallRecording` action.

The following example stops recording for the `call-id-1` call leg.

```
{
    "SchemaVersion": "1.0",
    "Actions":[
        {
            "Type": "StopCallRecording",
            "Parameters": {
                "CallId": "call-id-1"
            }
        }
    ]
}
```

CallId

Description – `CallId` of participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – Yes

Default value – None

See a working example on GitHub: https://github.com/aws-samples/amazon-chime-sma-on-demand-recording

PauseCallRecording

The `PauseCallRecording` action pauses recording of a call leg. Each time you pause a recording, the recording captures a tone that indicates the pause. When you pause, the recording continues, but it only captures silence. Pausing the recording does not affect the total duration of the recording. You can pause and resume recording as often as needed.

The following example pauses recording.

```
{
    "SchemaVersion": "1.0",
    "Actions":[
        {
            "Type": "PauseCallRecording",
            "Parameters": {
                "CallId": "call-id-1"
            }
        }
    ]
}
```

CallId

Description – `CallId` of participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – Yes

Default value – None

See a working example on GitHub: https://github.com/aws-samples/amazon-chime-sma-on-demand-recording

ResumeCallRecording

The `ResumeCallRecording` action resumes the recording of a call leg. Before the recording restarts, a brief tone is played. You can pause and resume a recording multiple times for the duration of the call leg.

The following example resumes recording.

```
{
    "SchemaVersion": "1.0",
    "Actions":[
        {
            "Type": "ResumeCallRecording",
            "Parameters": {
                "CallId": "call-id-1"
            }
        }
    ]
}
```

CallId

Description – `CallId` of participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – Yes

Default value – None

See a working example on GitHub: https://github.com/aws-samples/amazon-chime-sma-on-demand-recording

CallAndBridge

Creates an outbound call to a PSTN phone number, or to a SIP trunk configured as an Amazon Chime SDK Voice Connector or Amazon Chime SDK Voice Connector Group, and then bridges it with an existing call leg. You use `PSTN` when calling a phone number, and `AWS` when calling a SIP trunk.

An existing call leg can be an outbound call leg created by using the CreateSIPMediaApplicationCall API, or an inbound leg created by a SIP rule that invokes the AWS Lambda function with a `NewInboundCall` event. When you implement a `CallAndBridge` action to a Voice Connector or Voice Connector Group endpoint, you must specify the Amazon Resource Number (ARN) of the Voice Connector or Voice Connector Group.

You can also add custom SIP headers to outbound call legs and AWS Lambda functions. Custom headers allow you to pass values such as floor numbers and zip codes. For more information about custom headers, refer to Using SIP headers.

The following example code shows a typical action that bridges to a PSTN endpoint.

```
{
    "SchemaVersion": "1.0",
    "Actions": [{
            "Type": "CallAndBridge",
            "Parameters": {
                "CallTimeoutSeconds": 30,
                "CallerIdNumber": "e164PhoneNumber", // required
                "Endpoints": [{
                    "BridgeEndpointType": "PSTN", // required
                    "Uri": "e164PhoneNumber", // required
                }],
            }
        }
    ]
}
```

The following example shows a typical action that uses a Voice Connector or Voice Connector Group, plus a custom SIP header.

```
{
   "SchemaVersion":"1.0",
   "Actions":[
      {
         "Type":"CallAndBridge",
         "Parameters":{
            "CallTimeoutSeconds":30,
            "CallerIdNumber": "e164PhoneNumber", // required
            "RingbackTone": { // optional
                    "Type": "S3",
                    "BucketName": "s3_bucket_name",
```

```
                    "Key": "audio_file_name"
                },
            "Endpoints":[
               {
                  "BridgeEndpointType":"AWS", // enum type, required

                  "Arn":"arn:aws:chime:us-
east-1:0123456789101:vc/abcdefg1hijklm2nopq3rs" //VC or VCG ARN, required for AWS
 endpoints
                  "Uri":"ValidString", // required, see description below
               }
            ],
            "SipHeaders": {
                "x-String":"String"
            }
         }
      }
   ]
}
```

CallTimeoutSeconds

Description – The interval before a call times out. The timer starts at call setup.

Allowed values – Between 1 and 120, inclusive

Required – No

Default value – 30

CallerIdNumber

Description – A number belonging to the customer, or the From number of the A Leg

Allowed values – A valid phone number in the E.164 format

Required – Yes

Default value – None

Endpoints

Description – The endpoints of a call

Allowed values:

- `BridgeEndpointType` – AWS for Voice Connectors and Voice Connector Groups, otherwise PSTN.
- `Arn` – The ARN of a Voice Connector or Voice Connector Group. Only required when you use AWS as the `BridgeEndpointType`.
- `Uri` – The URI value depends on the type of endpoint.

  For PSTN endpoints, the URI must be a valid E.164 phone number.

  For AWS endpoints, the URI value sets the `user` part of the `Request-URI`. You must use Augmented Backus-Naur Format. Required length: between 1 and 30, inclusive. Use the following values: `a-z, A-Z, 0-9, &, =, +, $, /, %, -, _, !, ~, *, (,), (.)`

  The host value of the `Request-URI` is derived from the Inbound routes of the target Voice Connector. The following example shows a `CallAndBridge` action with an AWS endpoint.

```
{
   "SchemaVersion":"1.0",
   "Actions":[
      {
         "Type":"CallAndBridge",
         "Parameters":{
            "CallTimeoutSeconds":30,
            "CallerIdNumber": "+18005550122",
            "Endpoints":[
               {
                  "BridgeEndpointType":"AWS",
                  "Arn":"arn:aws:chime:us-
east-1:0123456789101:vc/abcdefg1hijklm2nopq3rs",
                  "Uri":"5550"
               }
            ],
            "SipHeaders": {
                "x-String":"String"
            }
         }
      }
   ]
}
```

  For more information about Inbound routes and Voice Connectors, refer to Editing Amazon Chime SDK Voice Connector settings.

Required – Yes

Default value – None

SipHeaders

Description – Enables you to pass additional values. Use only with the `AWS` endpoint type.

Allowed values – Valid SIP header

Required – No

Default value – None

The following example shows a successful `CallAndBridge` action that uses a PSTN endpoint:

```
{
   "SchemaVersion": "1.0",
   "Sequence": 3,
   "InvocationEventType": "ACTION_SUCCESSFUL",
   "ActionData": {
      "Type": "CallAndBridge",
      "Parameters": {
         "CallTimeoutSeconds": 30,
         "CallerIdNumber": "e164PhoneNumber",
         "Endpoints":[
            {
               "BridgeEndpointType": "PSTN",
               "Uri": "e164PhoneNumber"
            }
         ],
         "CallId": "call-id-1"
      }
   },
   "CallDetails":{
      .....
      .....
      "Participants":[
         {
            "CallId": "call-id-1",
            "ParticipantTag": "LEG-A",
            .....
            "Status": "Connected"
         },
```

```
         {
            "CallId": "call-id-2",
            "ParticipantTag": "LEG-B",
            .....
            "Status": "Connected"
         }
      ]
   }
}
```

The following example shows a failed `CallAndBridge` action.

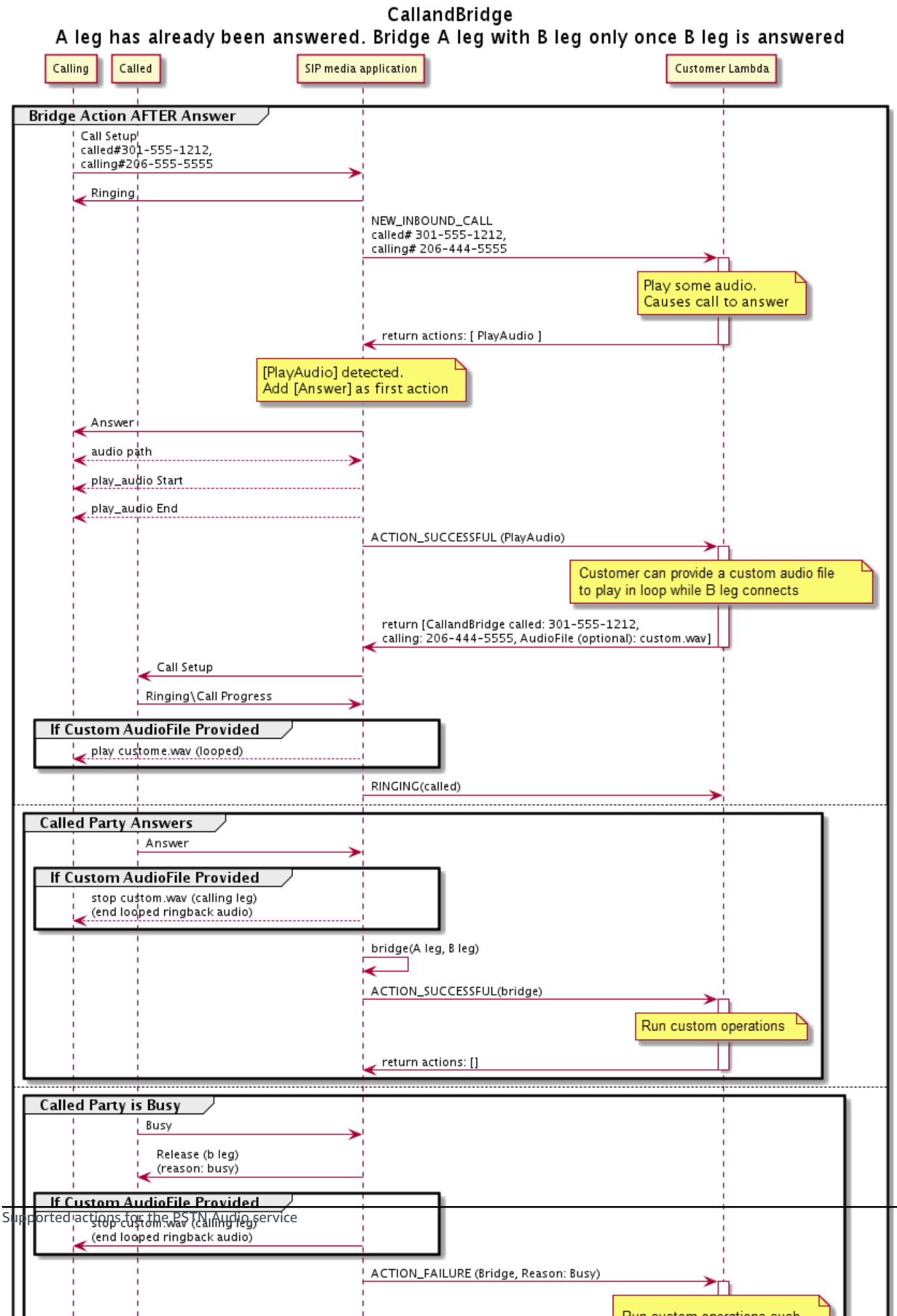
```
{
   "SchemaVersion": "1.0",
   "Sequence":2,
   "InvocationEventType": "ACTION_FAILED",
      "ActionData":{
      "Type": "CallAndBridge",
      "Parameters":{
         "CallTimeoutSeconds": 30,
         "CallerIdNumber": "e164PhoneNumber",
         "Endpoints": [
            {
               "BridgeEndpointType": "PSTN",
               "Uri": "e164PhoneNumber"
            }
         ],
         "CallId": "call-id-1"
      },
      "ErrorType": "CallNotAnswered",
      "ErrorMessage": "Call not answered"
   },
   "CallDetails":{
      .....
      .....
      "Participants":[
         {
            "CallId": "call-id-1",
            "ParticipantTag": "LEG-A",
            .....
         }
      ]
   }
```

```
}
```

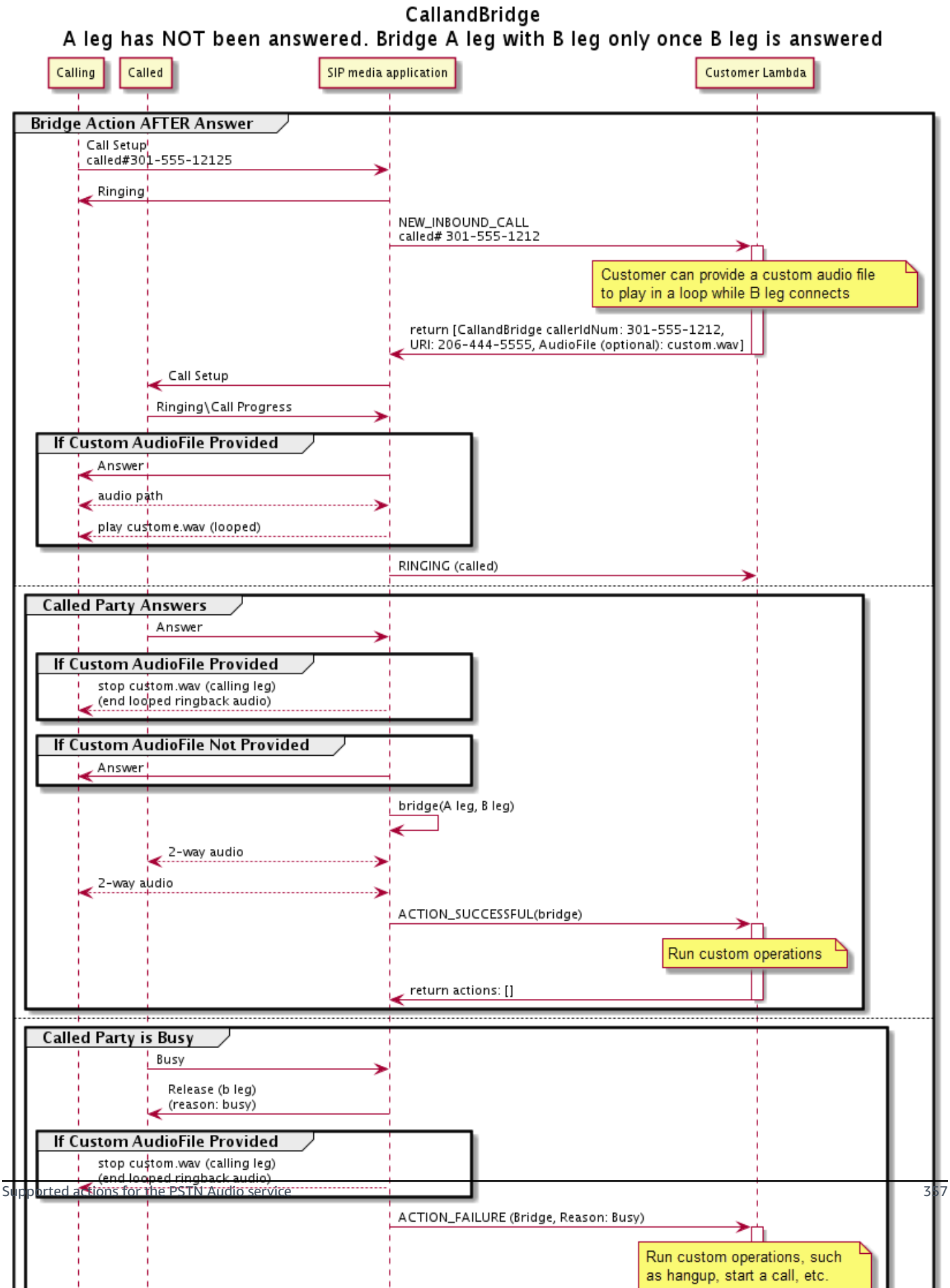
Call flows

The `CallAndBridge` action provides a different call signaling and audio experience for an existing call leg, depending on the parameters and whether the leg is connected.

The following diagram shows show the call flows with different parameters when an inbound call leg A is already connected.


CallandBridge
A leg has already been answered. Bridge A leg with B leg only once B leg is answered
Calling
Called
SIP media application
Customer Lambda
Bridge Action AFTER Answer
Call Setup
called#301-555-1212,
calling#206-555-5555
Ringing
NEW_INBOUND_CALL
called# 301-555-1212,
calling# 206-444-5555
Play some audio.
Causes call to answer
return actions: [PlayAudio]
[PlayAudio] detected.
Add [Answer] as first action
Answer
audio path
play_audio Start
play_audio End
ACTION_SUCCESSFUL (PlayAudio)
Customer can provide a custom audio file
to play in loop while B leg connects
return [CallandBridge called: 301-555-1212,
calling: 206-444-5555, AudioFile (optional): custom.wav]
Call Setup
Ringing\Call Progress
If Custom AudioFile Provided
play custome.wav (looped)
RINGING(called)
Called Party Answers
Answer
If Custom AudioFile Provided
stop custom.wav (calling leg)
(end looped ringback audio)
bridge(A leg, B leg)
ACTION_SUCCESSFUL(bridge)
Run custom operations
return actions: []
Called Party is Busy
Busy
Release (b leg)
(reason: busy)
If Custom AudioFile Provided
stop custom.wav (calling leg)
(end looped ringback audio)
ACTION_FAILURE (Bridge, Reason: Busy)

The following diagram shows the call flow for an unanswered call.


CallandBridge
A leg has NOT been answered. Bridge A leg with B leg only once B leg is answered
Calling
Called
SIP media application
Customer Lambda
Bridge Action AFTER Answer
Call Setup
called#301-555-12125
Ringing
NEW_INBOUND_CALL
called# 301-555-1212
Customer can provide a custom audio file to play in a loop while B leg connects
return [CallandBridge callerIdNum: 301-555-1212, URI: 206-444-5555, AudioFile (optional): custom.wav]
Call Setup
Ringing\Call Progress
If Custom AudioFile Provided
Answer
audio path
play custome.wav (looped)
RINGING (called)
Called Party Answers
Answer
If Custom AudioFile Provided
stop custom.wav (calling leg)
(end looped ringback audio)
If Custom AudioFile Not Provided
Answer
bridge(A leg, B leg)
2-way audio
2-way audio
ACTION_SUCCESSFUL(bridge)
Run custom operations
return actions: []
Called Party is Busy
Busy
Release (b leg)
(reason: busy)
If Custom AudioFile Provided
stop custom.wav (calling leg)
(end looped ringback audio)
ACTION_FAILURE (Bridge, Reason: Busy)
Run custom operations, such as hangup, start a call, etc.

Additional Details

Remember these facts about the `CallAndBridge` action.

- `CallTimeoutSeconds` – This timer starts when the SIP invitation is sent on the B-Leg. You can set a desired target value, but this value can be ignored by upstream carriers.
- `CallerIdNumber` – This phone number must belong to the customer, or be the From number of an A-Leg.
- **Hang-up behavior and edge cases** – If one call leg hangs up, the other call leg does not automatically hang up the call. When a `Hangup` event is sent to the AWS Lambda function, the remaining leg must be disconnected independently. If a call leg is left hanging, the call is billed until it is hung up. For example, the following scenario may lead to unexpected charges:
  - You try to bridge to a destination phone number. The destination is busy and sends the call straight to voicemail. From the Audio Service's perspective, going to voicemail is an answered call. The A-Leg hangs up, but the B-Leg continues listening for the voicemail message. While the B-Leg listens, you get billed.
  - As a best practice, use the AWS Lambda function, or the party on the other end of the call, to hang up each call leg independently.
- **Billing** – You're billed for the following when using `CallAndBridge`:
  - Active call minutes for each call leg created (A-Leg, B-Leg, etc.) to the PSTN.
  - Audio Service usage minutes.

See working examples on GitHub:

- https://github.com/aws-samples/amazon-chime-sma-bridging
- https://github.com/aws-samples/amazon-chime-sma-call-forwarding
- https://github.com/aws-samples/amazon-chime-sma-on-demand-recording

Hangup

Sends a `Hangup` value with a `SipStatusCode` to any leg of a call.

When the Audio Service runs a `Hangup` action on a call leg:

- For a call with only one call leg, the SIP media application invokes the AWS Lambda function with a `HANGUP` event and ignores the response. The call is then disconnected.

- For a call leg (Leg A) that is bridged to another call leg (Leg B), if the `Hangup` action is associated with the bridged call leg (Leg B) then the PSTN audio service disconnects the bridged call leg, then invokes the Lambda function with a `HANGUP` event for leg B. The PSTN audio service then runs any actions returned from that Lambda invocation.
- For a call leg (Leg A) that is bridged to another call leg (Leg B), if the `Hangup` action is associated with the original call leg (Leg A), then the PSTN audio service disconnects the original call leg, then invokes the Lambda function with a `HANGUP` event for leg A. The PSTN audio service then runs any actions returned from that Lambda invocation.
- For a call leg that joined to a meeting using the `JoinMeeting` action, if the `Hangup` action is associated with the meeting leg (usually Leg B) then the caller disconnects from the meeting and receives an `ACTION_SUCCESSFUL` event for the `Hangup` action.

The following example shows a typical `Hangup` action.

```
{
    "Type": "Hangup",
    "Parameters": {
        "CallId": "call-id-1",
        "ParticipantTag": "LEG-A",
        "SipResponseCode": "0"
    }
}
```

CallId

Description – `CallId` of participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – No

Default value – None

ParticipantTag

Description – `ParticipantTag` of one of the connected participants in the `CallDetails`

Allowed values – `LEG-A` or `LEG-B`

Required – No

Default value – `ParticipantTag` of the invoked `callLeg` Ignored if you specify `CallId`

SipResponseCode

Description – Any of the supported SIP response codes

Allowed values – 480–Unavailable; 486–Busy; 0–Normal Termination

Required – No

Default value – 0

After a user ends a call, the SIP media application invokes an AWS Lambda function with the code listed in Ending a call.

See working examples on GitHub:

- https://github.com/aws-samples/amazon-chime-sma-bridging
- https://github.com/aws-samples/amazon-chime-sma-call-forwarding
- https://github.com/aws-samples/amazon-chime-sma-outbound-call-notifications
- https://github.com/aws-samples/amazon-chime-sma-on-demand-recording

JoinChimeMeeting

Join an Amazon Chime SDK meeting by providing the attendee join token. To do this, you make AWS SDK calls to the CreateMeeting and CreateAttendee APIs to get the token and pass it on in the action. See the following example.

> **Note**
>
> You can't run this action on a bridged call.

```
{
    "Type": "JoinChimeMeeting",
    "Parameters": {
```

```
        "JoinToken": "meeting-attendee-join-token",
        "CallId": "call-id-1",
        "ParticipantTag": "LEG-A",
        "MeetingId": "meeting-id"
    }
}
```

JoinToken

Description – A valid join token of the Amazon Chime SDK meeting attendee

Allowed values – Valid join token

Required – Yes

Default value – None

CallId

Description – `CallId` of participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – No

Default value – None

ParticipantTag

Description – `ParticipantTag` of one of the connected participants in the `CallDetails`

Allowed values – `LEG-A`

Required – No

Default value – `ParticipantTag` of the invoked `callLeg` Ignored if you specify `CallId`

MeetingId

Description – A valid Amazon Chime SDK meeting ID associated with the `JoinToken`. If the meeting was created using an API in the Amazon Chime namespace, the meeting ID is not required. If the meeting was created using an API in the Amazon Chime SDK Meetings

namespace, the meeting ID is required. The meeting is joined using the API endpoint used to create the meeting.

Allowed values – A valid meeting ID.

Required – No.

Default value – None.

The SIP media application always invokes an AWS Lambda function after running this action. It returns either the `ACTION_SUCCESSFUL` or `ACTION_FAILED` invocation event types. The following example shows a successful invocation event structure.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 4,
    "InvocationEvent": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type": "JoinChimeMeeting",
        "Parameters": {
            "JoinToken": "meeting-attendee-join-token",
            "CallId": "call-id-1"
            "ParticipantTag": "LEG-A"
        }
    }
    "CallDetails": {
        ...
    }
}
```

Error handling

When a validation error occurs while bridging a meeting, the SIP application calls its AWS Lambda function with one of the error messages shown in the following table.

Error	Message	Reason
`InvalidActionParameter`	`JoinToken` parameter value is invalid.	Any of the action's other parameters is invalid or missing.

Error	Message	Reason
`SystemException`	System error while running action.	Another type of system error occurred while running the action.

The following example shows a typical failure event.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 3,
    "InvocationEvent": "ACTION_FAILED",
    "ActionData": {
        "Type": "JoinChimeMeeting",
        "Parameters": {
            "JoinToken": "meeting-attendee-join-token",
            "CallId": "call-id-1",
            "ParticipantTag": "LEG-A"
        },
        "Error": "ErrorJoiningMeeting: Error while joining meeting."
    }
    "CallDetails": {
        ...
    }
}
```

See a working example on GitHub: https://github.com/aws-samples/amazon-chime-sma-update-call

ModifyChimeMeetingAttendee (muting and unmuting audio)

Allows the SIP media application to modify the status of a telephony attendee by providing the Amazon Chime SDK meeting ID and attendee list.

> **Note**
>
> This action currently supports mute and unmute operations on telephony attendees. Also, the user must be joined into a meeting using the `JoinChimeMeeting` action. This action can be performed on a `participantTag=“LEG-B”`, or a corresponding `CallId`.

This action only applies to the callLeg that joins from the SIP media application to "+*13605550122*", LEG-B, or the leg joined from the SIP media application to the meeting.

```
{
"SchemaVersion": "1.0",
  "Actions": [
    {
      "Type" : "ModifyChimeMeetingAttendees",
      "Parameters" : {
        "Operation": "Mute",
        "MeetingId": "meeting-id",
        "CallId": "call-id",
        "ParticipantTag": LEG-B",
        "AttendeeList": ["attendee-id-1", "attendee-id-2"]
      }
    }
  ]
}
```

Operation

Description – The operation to perform on the list of attendees

Allowed values – Mute, Unmute

Required – Yes

Default value – None

MeetingId

Description – The ID of the meeting to which the attendees belong

Allowed values – A valid meeting ID. The person muting or unmuting must also belong to the meeting.

Required – Yes

Default value – None

CallId

Description – The ID of the meeting to which the attendees belong

Allowed values – A valid call ID.

Required – No

Default value – None

ParticipantTag

Description – The tag assigned to the attendee.

Allowed values – A valid tag.

Required – No

Default value – None

AttendeeList

Description – List of attendee IDs to mute or unmute

Allowed values – A list of valid attendee IDs

Required – Yes

Default value – None, maximum of 100

After running this action, Audio Service always invoke an AWS Lambda function with the `ACTION_SUCCESSFUL` or `ACTION_FAILED` invocation event type. The following example code shows a typical `ACTION_SUCCESSFUL` invocation event.

```
{
    "SchemaVersion": "1.0",
    "Sequence": INTEGER,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type" : "ModifyChimeMeetingAttendees",
        "Parameters" : {
            "Operation": "Mute",
            "MeetingId": "meeting-id",
            "CallId": "call-id",
            "ParticipantTag": "LEG-B",
            "AttendeeList": ["attendee-id-1", "attendee-id-2"]
        }
    }
    "CallDetails": {
        ...
    }
```

```
}
```

Error handling

In cases of invalid instruction parameters or API failures, SIP media applications call an AWS Lambda function with the error message specific to the failed instruction or API.

Error	Message	Reason
`InvalidActionParameter`	The `ModifyChimeMeetingAttendees Operation` parameter value is invalid	The `Operation` value must be Mute or Unmute.
	Meeting ID parameter value is invalid.	Meeting ID is empty.
	Attendee List parameter value is invalid.	The Attendee ID list is empty, or it exceeds the maximum of 100.
	Invalid action on the call.	The call isn't bridged.
	Call is not connected to Chime Meeting.	The attendee is not connected to a Chime Meeting.
	One or more attendees are not part of this meeting. All attendees must be part of this meeting.	The attendee is not authorized to modify attendees in the meeting.
`SystemException`	System error while running action.	A system error occurred while running an action.

The following example code shows a typical failure event:

```
{
    "SchemaVersion": "1.0",
    "Sequence": INTEGER,
```

```
    "InvocationEventType": "ACTION_FAILED",
    "ActionData": {
        "Type" : "ModifyChimeMeetingAttendees",
        "Parameters" : {
            "Operation": "Mute",
            "MeetingId": "meeting-id",
            "CallId": "call-id",
            "ParticipantTag": "LEG-B",
            "AttendeeList": ["attendee-id-1", "attendee-id-2"]
        },
        "ErrorType": "",
        "ErrorMessage": "",
        "ErrorList": []
    }
    "CallDetails": {
        ...
    }
}
```

See working examples on GitHub:

- https://github.com/aws-samples/amazon-chime-sma-bridging.
- https://github.com/aws-samples/amazon-chime-sma-update-call

Pause

Pause a call for a specified time.

```
{
    "Type": "Pause",
    "Parameters": {
        "CallId": "call-id-1",
        "ParticipantTag": "LEG-A",
        "DurationInMilliseconds": "3000"
    }
}
```

CallId

Description – `CallId` of participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – No

Default value – None

ParticipantTag

Description – `ParticipantTag` of one of the connected participants in the `CallDetails`

Allowed values – `LEG-A` or `LEG-B`

Required – No

Default value – `ParticipantTag` of the invoked `callLeg` Ignored if you specify `CallId`

DurationInMilliseconds

Description – Duration of the pause, in milliseconds

Allowed values – An integer >0

Required – Yes

Default value – None

See working examples on GitHub:

- https://github.com/aws-samples/amazon-chime-sma-outbound-call-notifications
- https://github.com/aws-samples/amazon-chime-sma-on-demand-recording

PlayAudio

Play an audio file on any leg of a call. The audio can be repeated any number of times. The in-progress audio can be terminated using the DTMF digits set in the `PlaybackTerminators`.

Currently, Amazon Chime SDK only supports playing audio files from the Amazon Simple Storage Service (Amazon S3) bucket. The S3 bucket must belong to the same AWS account as the SIP media application. In addition, you must give the `s3:GetObject` permission to the Amazon Chime SDK Voice Connector service principal. You can do that by using the S3 console or the command-line interface (CLI).

The following code example shows a typical bucket policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SMARead",
            "Effect": "Allow",
            "Principal": {
                "Service": "voiceconnector.chime.amazonaws.com"
            },
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::bucket-name/*",
                "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "aws-account-id"
                }
            }
        }
    ]
}
```

The Audio Service reads and writes to your S3 bucket on behalf of your Sip Media Application. To avoid the confused deputy problem, you can restrict S3 bucket access to a single SIP media application.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SMARead",
            "Effect": "Allow",
            "Principal": {
                "Service": "voiceconnector.chime.amazonaws.com"
            },
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::bucket-name/*",
                "Condition": {
                "StringEquals": {
```

```
                    "aws:SourceAccount": "aws-account-id",
                    "aws:SourceArn": "arn:aws:chime:region:aws-account-id:sma/sip-
media-application-id"
                }
            }
        }
    ]
}
```

The following code example shows a typical action.

```
{
    "Type": "PlayAudio",
    "Parameters": {
        "CallId": "call-id-1",
        "ParticipantTag": "LEG-A",
        "PlaybackTerminators": ["1", "8", "#"],
        "Repeat": "5",
        "AudioSource": {
            "Type": "S3",
            "BucketName": "valid-S3-bucket-name",
            "Key": "wave-file.wav"
        }
    }
}
```

CallID

Description – `CallId` of participant in the `CallDetails`.

Allowed values – A valid call ID.

Required – No, if `ParticipantTag` is present.

Default value – None.

ParticipantTag

Description – `ParticipantTag` of one of the connected participants in the `CallDetails`.

Allowed values – `LEG-A` or `LEG-B`.

Required – No, if `CallId` is present.

Default value – `ParticipantTag` of the invoked `callLeg`. Ignored if you specify `CallId`.

PlaybackTerminator

Description – Terminates in-progress audio by using DTMF input from the user

Allowed values – An array of the following values; “0”, ”1”, ”2”, ”3”, ”4”, ”5”, ”6”, ”7”, ”8”, ”9”, ”#”, ”*”

Required – No

Default value – None

Repeat

Description – Repeats the audio the specified number of times

Allowed values – An integer greater than zero

Required – No

Default value – 1

AudioSource.Type

Description – Type of source for audio file.

Allowed values – S3.

Required – Yes.

Default value – None.

AudioSource.BucketName

Description – For S3 source types, the S3 bucket must belong to the same AWS account as the SIP application. The bucket must have access to the Amazon Chime SDK Voice Connector service principal, which is voiceconnector.chime.amazonaws.com.

Allowed values – A valid S3 bucket for which Amazon Chime SDK has access to the `s3:GetObject` action.

Required – Yes.

Default value – None.

AudioSource.key

Description – For S3 source types, the file name from the S3 bucket specified in the `AudioSource.BucketName` attribute.

Allowed values – A valid audio file.

Required – Yes.

Default value – None.

The SIP media application tries to play the audio from the source URL. You can use raw, uncompressed PCM .wav files of no more than 50 MB in size. Amazon Chime SDK recommends 8 KHz mono.

When the last instruction in a dialplan is `PlayAudio` and the file finishes playback, or if a user stops playback with a key press, the application invokes the AWS Lambda function with the event shown in the following example.

```
{
    "SchemaVersion": "1.0",
    "Sequence": INTEGER,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type": "PlayAudio",
        "Parameters" : {
            "CallId": "call-id-1",
            "AudioSource": {
                "Type": "S3",
                "BucketName": "valid-S3-bucket-name",
                "Key": "wave-file.wav",
         }
     }
}
```

After a terminating digit stops the audio, it won't be repeated.

Error handling

When the validation file contains errors, or an error occurs while running an action, the SIP media application calls an AWS Lambda function with the appropriate error code.

Error	Message	Reason
`InvalidAudioSource`	The audio source parameter is invalid.	This error can happen for multiple reasons. For example, the SIP media application cannot access the file due to permission issues, or issues with the URL. Or, the audio file may fail validation due to format, duration, size, and so on.
`SystemException`	System error while running action.	Another system error occurred while running the action.
`InvalidActionParameter`	CallId or ParticipantTag parameter for action is invalid.	The action contains an invalid parameter.

The following code example shows a typical invocation failure.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 2,
    "InvocationEventType": "ACTION_FAILED",
    "ActionData": {
        "Type": "PlayAudio",
        "Parameters" : {
            "CallId": "call-id-1",
            "AudioSource": {
                "Type": "S3",
                "BucketName": "bucket-name",
                "Key": "audio-file.wav"
            },
        },
        "ErrorType": "InvalidAudioSource",
        "ErrorMessage": "Audio Source parameter value is invalid."
    }
```

```
    "CallDetails": {
        ...
    }
}
```

See working examples on GitHub:

- https://github.com/aws-samples/amazon-chime-sma-bridging.
- https://github.com/aws-samples/amazon-chime-sma-call-forwarding
- https://github.com/aws-samples/amazon-chime-sma-outbound-call-notifications
- https://github.com/aws-samples/amazon-chime-sma-on-demand-recording
- https://github.com/aws-samples/amazon-chime-sma-update-call

PlayAudioAndGetDigits

Plays audio and gathers DTMF digits. If a failure occurs, such as a user not entering the correct number of DTMF digits, the action plays the "failure" audio and then replays the main audio until the SIP media application exhausts the number of attempts defined in the `Repeat` parameter.

You must play audio files from the S3 bucket. The S3 bucket must belong to the same AWS account as the SIP media application. In addition, you must give the `s3:GetObject` permission to the Amazon Chime SDK Voice Connector service principal, `voiceconnector.chime.amazonaws.com`. You can use the S3 console or the CLI to do that.

The following code example shows a typical S3 bucket policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SMARead",
            "Effect": "Allow",
            "Principal": {
                "Service": "voiceconnector.chime.amazonaws.com"
            },
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::bucket-name/*",
                "Condition": {
```

```
                "StringEquals": {
                    "aws:SourceAccount": "aws-account-id"
                }
            }
        }
    ]
}
```

The Audio Service reads and writes to your S3 bucket on behalf of your Sip Media Application. To avoid the confused deputy problem, you can restrict S3 bucket access to a single SIP media application.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SMARead",
            "Effect": "Allow",
            "Principal": {
                "Service": "voiceconnector.chime.amazonaws.com"
            },
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::bucket-name/*",
                "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "aws-account-id",
                    "aws:SourceArn": "arn:aws:chime:region:aws-account-id:sma/sip-
media-application-id"
                }
            }
        }
    ]
}
```

The following example shows a typical `PlayAudioAndGetDigits` action.

```
{
    "Type" : "PlayAudioAndGetDigits",
    "Parameters" : {
        "CallId": "call-id-1",
```

```
        "ParticipantTag": "LEG-A"
        "InputDigitsRegex": "^\d{2}#$",
        "AudioSource": {
            "Type": "S3",
            "BucketName": "bucket-name",
            "Key": "audio-file-1.wav"
        },
        "FailureAudioSource": {
            "Type": "S3",
            "BucketName": "bucket-name",
            "Key": "audio-file-failure.wav"
        },
        "MinNumberOfDigits": 3,
        "MaxNumberOfDigits": 5,
        "TerminatorDigits": ["#"],
        "InBetweenDigitsDurationInMilliseconds": 5000,
        "Repeat": 3,
        "RepeatDurationInMilliseconds": 10000
    }
}
```

CallId

Description – `CallId` of participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – No

Default value – None

ParticipantTag

Description – `ParticipantTag` of one of the connected participants in the `CallDetails`

Allowed values – `LEG-A` or `LEG-B`

Required – No

Default value – `ParticipantTag` of the invoked `callLeg` Ignored if you specify `CallId`

InputDigitsRegex

Description – A regular expression pattern

Allowed values – A valid regular expression pattern

Required – No

Default value – None

AudioSource.Type

Description – Type of source for the audio file type

Allowed values – An S3 bucket

Required – Yes

Default value – `"S3"`

AudioSource.BucketName

Description – For S3 `AudioSource.Type` values, the S3 bucket must belong to the same AWS account as the SIP media application. The bucket S3 must have access to the Amazon Chime SDK Voice Connector service principal, `voiceconnector.chime.amazonaws.com`.

Allowed values – A valid S3 bucket for which Amazon Chime SDK has `s3:GetObject` actions access.

Required – Yes

Default value – None

AudioSource.Key

Description – The key name of the audio object in the `AudioSource.BucketName` S3 bucket.

Allowed values – Valid audio files

Required – Yes

Default value – None

FailureAudioSource.Type

Description – The key name of the audio object in the `FailureAudioSource.BucketName` S3 bucket.

Allowed values – S3

Required – Yes

Default value – None

FailureAudioSource.BucketName

Description – For S3 source types, the S3 bucket must belong to the same AWS account as the SIP media application. The Amazon Chime SDK Voice Connector service principal, `voiceconnector.chime.amazonaws.com`, must have access to the S3 bucket.

Allowed values – A valid S3 bucket for which Amazon Chime SDK has `s3:GetObject` actions access.

Required – Yes

Default value – None

FailureAudioSource.Key

Description – The key name of the audio object in the `FailureAudioSource.BucketName` S3 bucket.

Allowed values – Valid audio files

Required – Yes

Default value – None

MinNumberOfDigits

Description – The minimum number of digits to capture before timing out or playing "call failed" audio.

Allowed values – >=0

Required – No

Default value – 0

MaxNumberOfDigits

Description – The maximum number of digits to capture before stopping without a terminating digit.

Allowed values – >`MinNumberOfDigits`

Required – No

Default value – 128

TerminatorDigits

Description – Digits used to end input if the user enters less than the `MaxNumberOfDigits`

Allowed values – Any one of these digits: 0123456789#*

Required – No

Default value – #

InBetweenDigitsDurationInMilliseconds

Description – The wait time in milliseconds between digit inputs before playing `FailureAudio`.

Allowed values – >0

Required – No

Default value – If not specified, defaults to the `RepeatDurationInMilliseconds` value.

Repeat

Description – Total number of attempts to get digits

Allowed values – >0

Required – No

Default value – 1

RepeatDurationInMilliseconds

Description – Time in milliseconds to wait between `Repeat` attempts

Allowed values – >0

Required – Yes

Default value – None

The SIP media application always invokes its AWS Lambda function after running the `PlayAudioAndGetDigits` action, with an `ACTION_SUCCESSFUL` or `ACTION_FAILED` invocation event type. When the application successfully gathers digits, it sets the `ReceivedDigits` value in the `ActionData` object. The following example shows the invocation event structure of that AWS Lambda function.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 3,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type": "PlayAudioAndGetDigits",
        "Parameters" : {
            "CallId": "call-id-1",
            "ParticipantTag": "LEG-A",
            "InputDigitsRegex": "^\d{2}#$",
            "AudioSource": {
                "Type": "S3",
                "BucketName": "bucket-name",
                "Key": "audio-file-1.wav"
            },
            "FailureAudioSource": {
                "Type": "S3",
                "BucketName": "bucket-name",
                "Key": "audio-file-failure.wav"
            },
            "MinNumberOfDigits": 3,
            "MaxNumberOfDigits": 5,
            "TerminatorDigits": ["#"],
            "InBetweenDigitsDurationInMilliseconds": 5000,
            "Repeat": 3,
            "RepeatDurationInMilliseconds": 10000
        },
        "ErrorType": "InvalidAudioSource",
        "ErrorMessage": "Audio Source parameter value is invalid."
    },
        "ReceivedDigits": "1234"
    },
    "CallDetails": {
        ...
```

```
    }
}
```

Error handling

When a validation error occurs, the SIP media application calls the AWS Lambda function with the corresponding error message. The following table lists the possible error messages.

Error	Message	Reason
`InvalidAudioSource`	Audio source parameter value is invalid.	This error can happen for multiple reasons. For example, the SIP media application cannot access the file due to permission issues, or issues with the S3 bucket. Or, the audio file may fail validation due to duration, size, or unsupported format.
`InvalidActionParameter`	`CallId` or `ParticipantTag` parameter for the action is invalid.	A `CallId`, `ParticipantTag`, or other parameter is not valid.
`SystemException`	System error while running the action.	A system error occurred while running the action.

When the action fails to collect the number of specified digits due to a timeout or too many retries, the SIP media application invokes the AWS Lambda function with the `ACTION_FAILED` invocation event type.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 4,
    "InvocationEventType": "ACTION_FAILED",
    "ActionData": {
        "Type": "PlayAudioAndGetDigits",
        "Parameters" : {
            "CallId": "call-id-1",
```

```
            "ParticipantTag": "LEG-A",
            "InputDigitsRegex": "^\d{2}#$",
            "AudioSource": {
                "Type": "S3",
                "BucketName": "bucket-name",
                "Key": "audio-file-1.wav"
            },
            "FailureAudioSource": {
                "Type": "S3",
                "BucketName": "bucket-name",
                "Key": "audio-file-failure.wav"
            },
            "MinNumberOfDigits": 3,
            "MaxNumberOfDigits": 5,
            "TerminatorDigits": ["#"],
            "InBetweenDigitsDurationInMilliseconds": 5000,
            "Repeat": 3,
            "RepeatDurationInMilliseconds": 10000
        },
        "ErrorType": "InvalidAudioSource",
        "ErrorMessage": "Audio Source parameter value is invalid."
    }
    "CallDetails": {
        ...
    }
}
```

See working examples on GitHub:

- https://github.com/aws-samples/amazon-chime-sma-bridging.
- https://github.com/aws-samples/amazon-chime-sma-update-call

ReceiveDigits

When a user enters digits that match the regular expression pattern specified in this action, the SIP media application invokes the AWS Lambda function.

```
{
    "Type": "ReceiveDigits",
    "Parameters": {
        "CallId": "call-id-1",
        "ParticipantTag": "LEG-A",
```

```
        "InputDigitsRegex": "^\d{2}#$",
        "InBetweenDigitsDurationInMilliseconds": 1000,
        "FlushDigitsDurationInMilliseconds": 10000
    }
}
```

CallId

Description – `CallId` of participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – No

Default value – None

ParticipantTag

Description – `ParticipantTag` of one of the connected participants in the `CallDetails`

Allowed values – `LEG-A` or `LEG-B`

Required – No

Default value – `ParticipantTag` of the invoked `callLeg` Ignored if you specify `CallId`

InputDigitsRegex

Description – A regular expression pattern

Allowed values – A valid regular expression pattern

Required – Yes

Default value – None

InBetweenDigitsDurationInMilliseconds

Description – Interval between digits before checking to see if the input matches the regular expression pattern

Allowed values – Duration in milliseconds

Required – Yes

Default value – None

FlushDigitsDurationInMilliseconds

Description – Interval after which received DTMF digits are flushed and sent to the AWS Lambda function. If the SIP media application receives a new digit after the interval ends, the timer starts again.

Allowed values – `InBetweenDigitsDurationInMilliseconds`

Required – Yes

Default value – None

The SIP media application discards DTMF digits for the duration of a call until it receives a new `ReceiveDigits` action. The `FlushDigitsDurationInMilliseconds` interval starts when the SIP media application receives the first DTMF digit. If the user enters the correct digits before the interval expires, the SIP media application invokes the AWS Lambda function described in Receiving caller input.

If the user input doesn't match the regular expression pattern, the SIP media application repeats the "failure" audio file message until the application exhausts the repeat count or the user inputs valid digits.

See working examples on GitHub:

- https://github.com/aws-samples/amazon-chime-sma-outbound-call-notifications
- https://github.com/aws-samples/amazon-chime-sma-on-demand-recording
- https://github.com/aws-samples/amazon-chime-sma-update-call

RecordAudio

Allows the SIP media application to record media from a given call ID. For example, a voice mail application and meeting-participant announcements. The application records until it reaches the duration that you set, or when a user presses one of the `RecordingTerminators`, or when the application detects silence. In those cases, the action tells your application to put the resulting media file into the specified S3 bucket. The S3 bucket must belong to the same AWS account as the SIP media application. In addition, the action must give `s3:PutObject` and `s3:PutObjectAcl` permission to the Amazon Chime SDK Voice Connector service principal, Amazon Chime SDK Voice Connector service principal, `voiceconnector.chime.amazonaws.com`.

> **Note**
>
> Recordings made using this feature may be subject to laws or regulations regarding the recording of electronic communications. It is your and your end users' responsibility to comply with all applicable laws regarding the recording, including properly notifying all participants in a recorded session or communication that the session or communication is being recorded, and obtaining their consent.

The following example gives the `s3:PutObject` and `s3:PutObjectAcl` permission to the Amazon Chime SDK Voice Connector service principal.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SMARead",
            "Effect": "Allow",
            "Principal": {
                "Service": "voiceconnector.chime.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::bucket-name/*"
        }
    ]
}
```

The following example stops recording when the caller presses the pound key (#), or 10 seconds elapse with no activity, or the caller remains silent for 3 seconds, and it writes the resulting media file into the location defined by `RecordingDestination` parameter.

> **Note**
>
> This example uses the `CallId` parameter. You can use the `ParticipantTag` parameter instead, but you can't use both.

```
{
    "Type": "RecordAudio",
    "Parameters": {
        "CallId": "call-id-1",
        "DurationInSeconds": "10",
        "SilenceDurationInSeconds": 3,
        "SilenceThreshold": 100,
        "RecordingTerminators": [
            "#"
        ],
        "RecordingDestination": {
            "Type": "S3",
            "BucketName": "valid-bucket-name",
            "Prefix": "valid-prefix-name"
        }
    }
}
```

CallId

Description – `CallId` of participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – No

Default value – None

ParticipantTag

Description – `ParticipantTag` of one of the connected participants in the `CallDetails`

Allowed values – `LEG-A` or `LEG-B`

Required – No

Default value – `ParticipantTag` of the invoked `callLeg` Ignored if you specify `CallId`

RecordingDestination.Type

Description – Type of destination. Only S3.

Allowed values – S3

Required – Yes

Default value – None

RecordingDestination.BucketName

Description – A valid S3 bucket name. The bucket must have access to the Amazon Chime SDK Voice Connector service principal, `voiceconnector.chime.amazonaws.com`.

Allowed values – A valid S3 bucket for which Amazon Chime SDK has access to the `s3:PutObject` and `s3:PutObjectAcl` actions.

Required – Yes

Default value – None

RecordingDestination.Prefix

Description – S3 prefix of recording file

Allowed values – A valid prefix name containing up to 979 safe characters. For more information about safe characters, refer to Safe characters in the Amazon Simple Storage Service User Guide.

Required – No

Default – None. If not specified, the recording are saved to the root of the S3 bucket.

DurationInSeconds

Description – The duration of the recording, in seconds

Allowed values – >0

Required – No

Default value – None

SilenceDurationInSeconds

Description – The duration of silence in seconds, after which the recording stops. If not specified, silence detection is disabled.

Allowed values – [1;1000]

Required – No

Default value – 200

SilenceThreshold

Description – Level of noise that is considered "silence." If you don't specify `SilenceDurationInSeconds`, this parameter is ignored.

Reference values (noise levels and thresholds to treat the noise as silence):

- 1—30dB or below, such as a quiet room
- 100—40-50 dB, such as a whisper or quiet office
- 200—60dB, such as a crowded office
- 1000—75 dB, such as a loud person or music

Allowed values – [1;1000]

Required – No

Default value – 200

RecordingTerminators

Description – Lists all the available recording terminators.

Allowed values – An array of single digits and symbols from [123456789*0#]

Required – Yes

Default value – None

Handling ACTION_SUCCESSFUL events

When the recording ends, the Amazon Chime SDK SIP media application calls the AWS Lambda function and passes to it the ACTION_SUCCESSFUL event, along with the invocation results.

```
{
    "SchemaVersion": "1.0",
    "Sequence": INTEGER,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type" : "RecordAudio",
        "Parameters": {
           ...
        },
```

```
        "RecordingDestination": {
            "Type": "S3",
            "BucketName": "valid-bucket-name",
            "Key": "valid-S3-key"
        },
        "RecordingTerminatorUsed":"#"
    },
    "CallDetails": {
        ...
    }
}
```

The `ACTION_SUCCESSFUL` event contains `ActionData`, which contains these fields:

Type

Description – The type of the action, `RecordAudio`.

Parameters

Description – The parameters of the action.

RecordingDestination.Type

Description – Type of destination. Only S3.

RecordingDestination.BucketName

Description – The S3 bucket that contains the recording file.

RecordingDestination.Key

Description – The S3 key of the recording file.

RecordingTerminatorUsed

Description – The terminator used to stop recording—one of the terminators passed in the `RecordingTerminators` parameter. If the recording stops after reaching maximum duration (`DurationInSeconds`) or because of silence (`SilenceDurationInSeconds`), this key-value pair is not included in the output.

Error handling

For validation errors, the SIP media application calls the AWS Lambda function with the appropriate error message. The following table lists the possible error messages.

Error	Message	Reason
`InvalidActionParameter`	`CallId` or `ParticipantTag` parameter for action is invalid. `DurationInSeconds` parameter value is invalid. `SilenceDurationInSeconds` parameter value is invalid. `SilenceThreshold` parameter value is invalid. `RecordingDestination` parameter value is invalid. Error occurred while uploading recording to S3 bucket.	Any parameter is invalid.
`SystemException`	System error while running an action.	Another type of system error occurred while running an action.

Handling ACTION_FAILED events

When the action fails to record the media on a call leg, the SIP media application invokes an AWS Lambda function with the `ACTION_FAILED` event type. See the following example.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 5,
    "InvocationEventType": "ACTION_FAILED",
    "ActionData": {
        "Type" : "RecordAudio",
        "Parameters": {
```

```
            ...
        },
        "ErrorType": "InvalidActionParameter",
        "ErrorMessage": "RecordingDestination parameter value is invalid."
    },
    "CallDetails": {
        ...
    }
}
```

See a working example on GitHub: https://github.com/aws-samples/amazon-chime-sma-bridging

SendDigits

Send up to 50 dual tone multi-frequency (DTMF) digits on any leg of a call. The signals can include the following:

- Numbers 0 thru 9
- Special characters star (*) and pound (#)
- Network control signals A, B, C, D
- The comma character (,). This signal adds a 0.5 second delay between the previous and next signals.

Topics

Using the SendDigits action

The following example shows a typical `SendDigits` action:

```
{
    "SchemaVersion": "1.0",
    "Actions":[
        {
            "Type": "SendDigits",
```

```
            "Parameters": {
                "CallId": "call-id-1", // required
                "Digits": ",,*1234,56,7890ABCD#", // required
                "ToneDurationInMilliseconds": 100 // optional
            }
        }
    ]
}
```

CallId

Description – The `CallId` of a participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – Yes

Default value – None

Digits

Description – The digits to be sent on the call leg that corresponds to the `CallId`

Allowed values – 0-9, *, #, A, B, C, D, comma (,)

Required – Yes

Default value – None

ToneDurationInMilliseconds

Description – The amount of time allowed, in milliseconds, to transmit each digit.

Allowed values – Any integer between 50 and 24000

Required – No

Default value – 250

Handling ACTION_SUCCESSFUL events

The following example shows a typical `ACTION_SUCCESSFUL` event for the `SendDigits` action.

```
{
```

```
    "SchemaVersion": "1.0",
    "Sequence": 3,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type": "SendDigits",
        "Parameters": {
            "Digits": "1,2A#",
            "ToneDurationInMilliseconds": 100,
            "CallId": "call-id-1"
        },
    "CallDetails": {
        ...
        }
    }
}
```

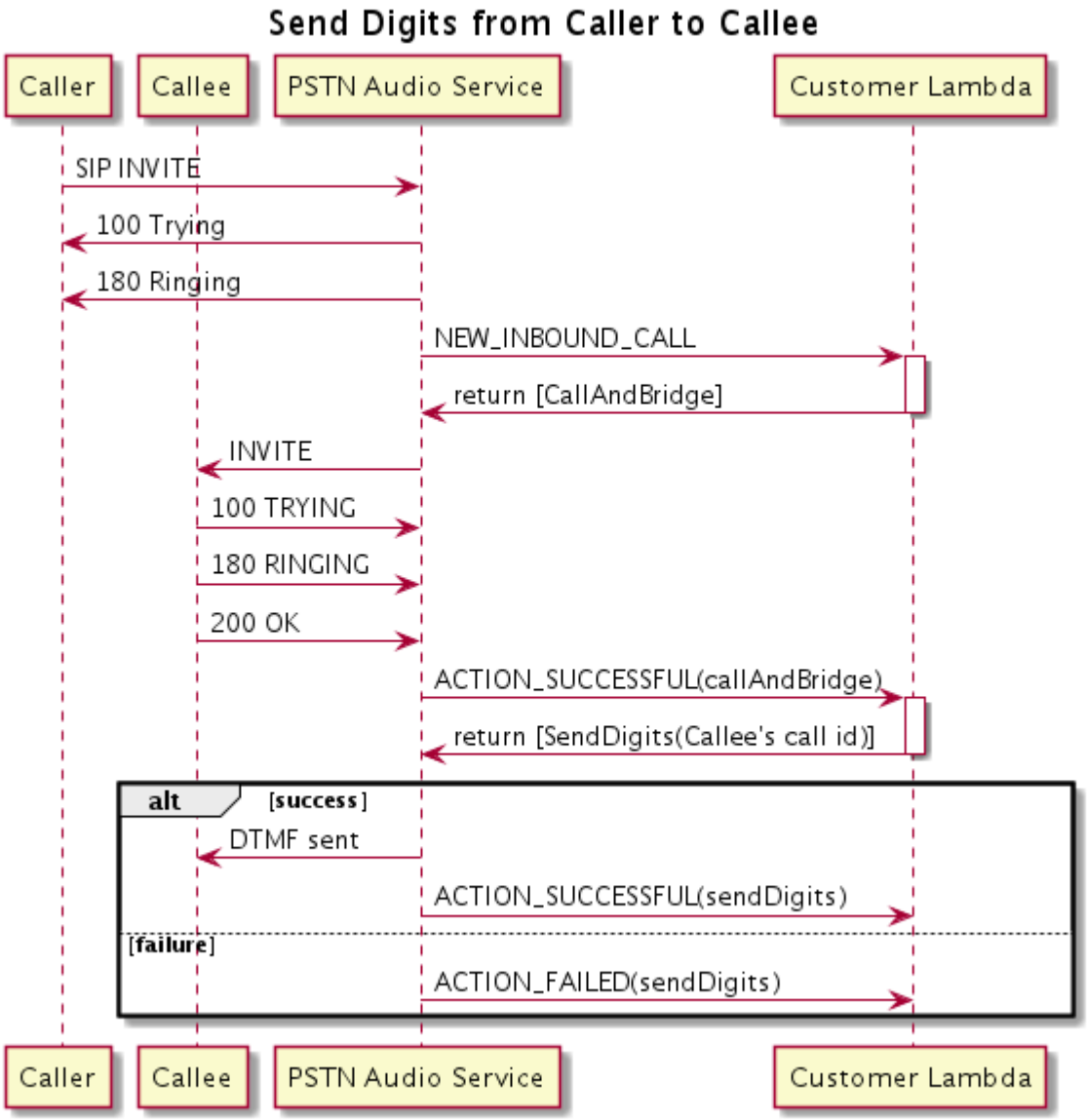
Handling ACTION_FAILED events

The following example shows a typical `ACTION_FAILED` event for the `SendDigits` action.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 3,
    "InvocationEventType": "ACTION_FAILED",
    "ActionData": {
        "Type": "SendDigits",
        "Parameters": {
            "Digits": "1,2A#",
            "ToneDurationInMilliseconds": 20000000,
            "CallId": "call-id-1"
        },
    "ErrorType": "InvalidActionParameter",
    "ErrorMessage": "ToneDuration parameter value is invalid."
    },
    "CallDetails": {
        ...
        }
    }
}
```

Call flow

The following diagram shows the program flow for sending digits from a caller to a callee.



Speak

You can play speech on any call leg by providing text. You can use plain text or Speech Synthesis Markup Language (SSML). SSML provides more control over how the Amazon Chime SDK generates speech by adding pauses, emphasizing certain words, or changing the speaking style.

The Amazon Chime SDK uses the Amazon Polly service to convert text-to-speech. Amazon Polly allows you to choose between either the standard or neural engine for improved speech quality. Amazon Polly supports more than 20 languages and 60 voices to customize your application's user experience. The Amazon Chime SDK provides speech features at no charge, but you do pay for using Amazon Polly. See the Amazon Polly pricing page or your billing dashboard for pricing information.

> **Important**
>
> Use of Amazon Polly is subject to the AWS Service Terms, including the terms specific to the AWS Machine Learning and Artificial Intelligence Services.

Topics

Using the Speak action

The following example shows a typical use of the `Speak` action.

```
{
    "SchemaVersion": "1.0",
    "Actions":[
        {
            "Type": "Speak",
            "Parameters": {
                "Text": "Hello, World!",        // required
                "CallId": "call-id-1",          // required
                "Engine": "neural",             // optional. Defaults to standard
                "LanguageCode": "en-US",        // optional
                "TextType": "text",             // optional
                "VoiceId": "Joanna"             // optional. Defaults to Joanna
            }
        }
    ]
}
```

CallId

Description – The `CallId` of participant in the `CallDetails` of the Lambda function invocation

Allowed values – A valid call ID

Required – Yes

Default value – None

Text

Description – Specifies the input text to synthesize into speech. If you specify `ssml` as the `TextType`, follow the SSML format for the input text.

Allowed values – String

Required – Yes

Default value – None

Engine

Description – Specifies the engine—standard or neural—to use when processing text for speech synthesis.

Allowed values – standard | neural

Required – No

Default value – standard

LanguageCode

Description – Specifies the language code. Only necessary if using a bilingual voice. If you use a bilingual voice without a language code, the bilingual voice's default language is used.

Allowed values – Amazon Polly language codes

Required – No

Default value – None

TextType

Description – Specifies the type of input text, plain text or SSML. If an input type is not specified, plain text is used as the default. For more information about SSML, see Generating Speech from SSML Documents in the *Amazon Polly Developer Guide*.

Allowed values – ssml | text

Required – No

Default value – None

VoiceId

Description – Specifies the ID of voice you want to use.

Allowed values – Amazon Polly voice IDs

Required – No

Default value – Joanna

Handling ACTION_SUCCESSFUL events

The following example shows a typical `ACTION_SUCCESSFUL` event for an action which synthesizes the text "Hello World" into speech, in English, using the Amazon Polly's `Joanna` voice.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 3,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
       "Type": "Speak",
       "Parameters": {
          "CallId": "call-id-1",
          "Engine":  "neural",
          "LanguageCode":  "en-US",
          "Text": "Hello World",
          "TextType":  "text",
          "VoiceId":  "Joanna"
       }
    },
    "CallDetails":{
       ...
    }
}
```

Handling ACTION_FAILED events

The following example shows a typical `ACTION_FAILED` event for the same event used in the previous example.

```
{
    "SchemaVersion": "1.0",
    "Sequence":2,
```

```
    "InvocationEventType": "ACTION_FAILED",
    "ActionData":{
       "Type": "Speak",
       "Parameters": {
          "CallId": "call-id-1",
          "Engine":  "neural",
          "LanguageCode":  "en-US",
          "Text": "Hello  World",
          "TextType":  "text",
          "VoiceId":  "Joanna"
       },
       "ErrorType": "SystemException",
       "ErrorMessage": "System error while running  action"
    },
    "CallDetails":{
       ...
    }
}
```

Error handling

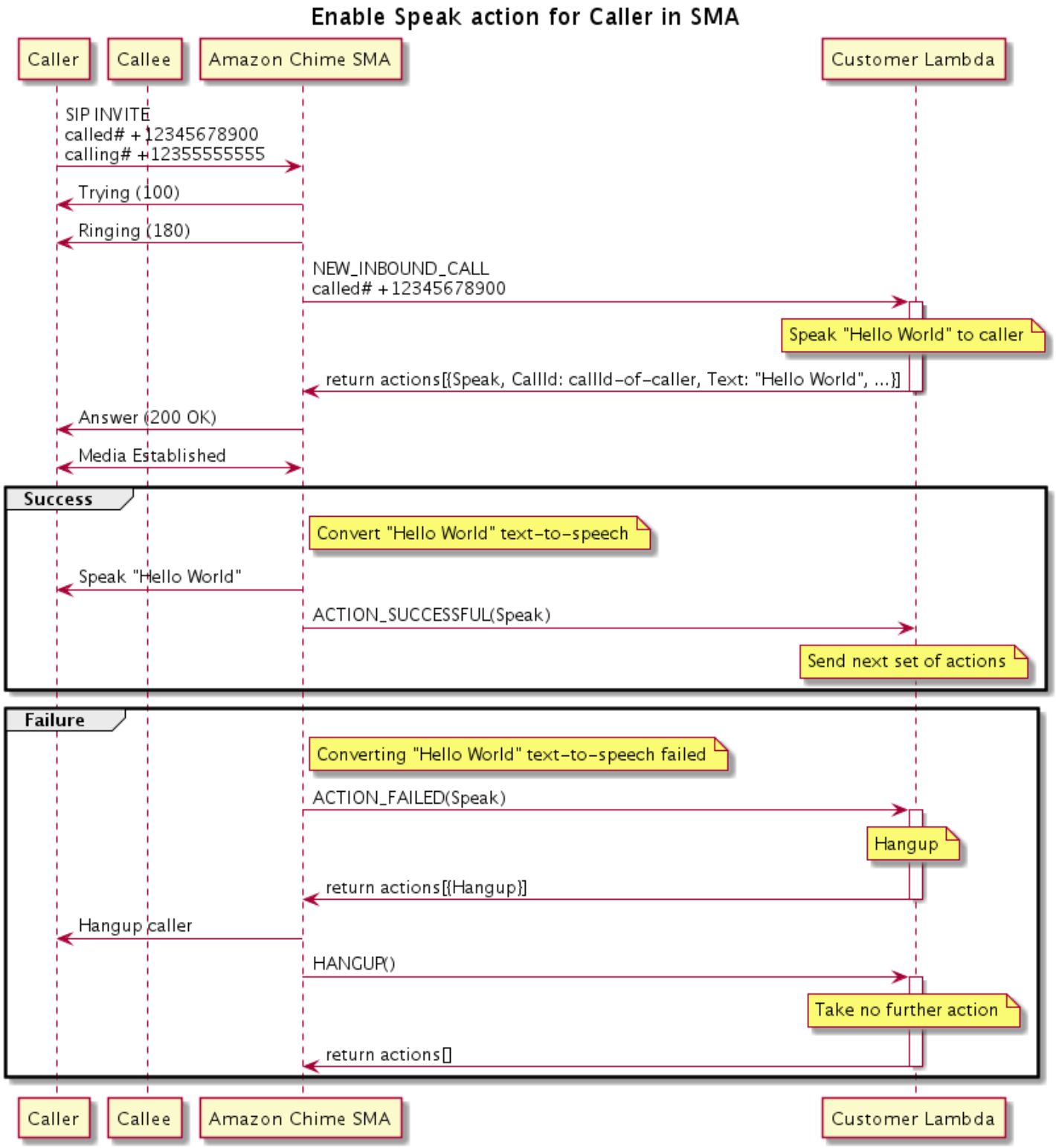
This table lists and describes the error messages thrown by the the `Speak` action.

Error	Message	Reason
`AccessDenied`	The `AWSServiceRoleForAmazonChimeVoiceConnector` service-linked role is not configured correctly.	The service-linked role used to make requests to Amazon Polly doesn't exist or is missing permissions. To resolve, see the steps in the Using the Amazon Chime SDK Voice Connector service-linked role section
`InvalidActionParameter`		There was an error validating the action parameters. See the SynthesizeSpeech API in the *Amazon Polly Developer Guide* for more information about parameters.

Error	Message	Reason
`ActionExecutionThrottled`	Amazon Polly is throttling the request to synthesize speech.	The request to Amazon Polly is returning a throttling exception. For more information about the Amazon Polly throttling limits, see https://docs.aws.amazon.com/polly/latest/dg/limits.html#limits-throttle .
`MissingRequiredActionParameter`	`Text` is a required parameter.	There action parameters must have a `Text` value
`MissingRequiredActionParameter`	`Text` is limited to 1,000 characters	The text exceeded the character limit.
`SystemException`	System error while running action.	A system error occurred while running the action.

Program flows

The following diagram shows the program flow that enables the `Speak` action for a caller. In this example, the caller hears text that



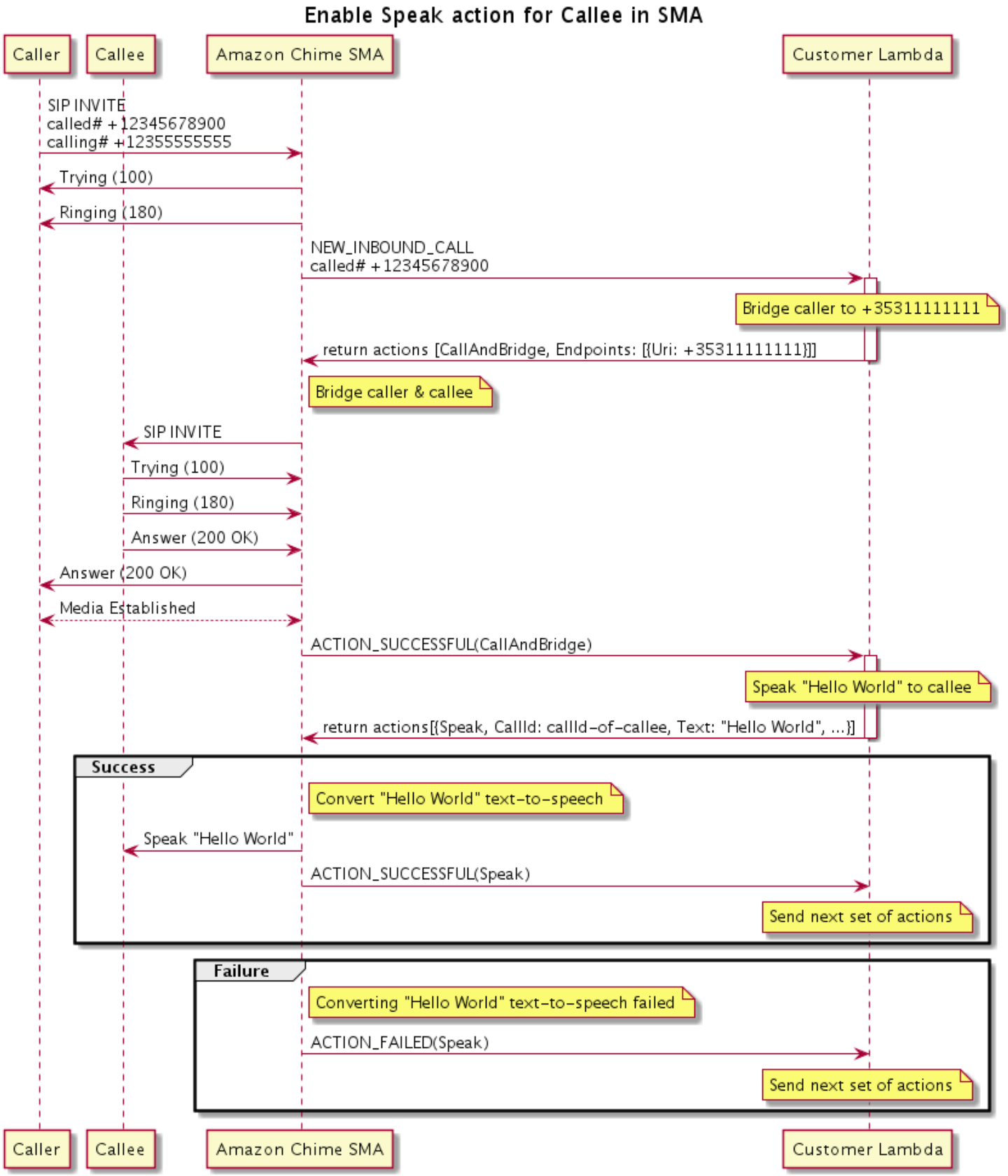
In the diagram

Using a soft phone, a caller enters a number registered to a SIP media application. The application uses the SIP `INVITE` method and sends the caller a `Trying (100)` response. That indicates that the next-hop server received the call request. The SIP application then uses `INVITE` to contact the endpoint. Once the connection is established, the applications sends `Ringing (180)` response to the caller, and alerting begins.

The SIP media application then sends a `NEW_INBOUND_CALL` event to the Lambda function, which responds with a `Speak` action that includes the caller's ID and the text that you want to convert into speech. The SIP application then sends a `200 (OK)` response to indicate that the call was answered. The protocol also enables the media.

If the `Speak` action succeeds and converts the text to speech, it returns an `ACTION_SUCCESSFUL` event to the SIP media application, which returns the next set of actions. If the action fails, the SIP media application sends an `ACTION_FAILED` event to the Lambda function, which responds with a set of `Hangup` actions. The application hangs up the caller and returns a `HANGUP` event to the Lambda function, which takes no further actions.

The following diagram shows the program flow than enables the `Speak` action for a callee.


Enable Speak action for Callee in SMA
Caller
Callee
Amazon Chime SMA
Customer Lambda
SIP INVITE
called# +12345678900
calling# +12355555555
Trying (100)
Ringing (180)
NEW_INBOUND_CALL
called# +12345678900
Bridge caller to +35311111111
return actions [CallAndBridge, Endpoints: [{Uri: +35311111111}]]
Bridge caller & callee
SIP INVITE
Trying (100)
Ringing (180)
Answer (200 OK)
Answer (200 OK)
Media Established
ACTION_SUCCESSFUL(CallAndBridge)
Speak "Hello World" to callee
return actions[{Speak, CallId: callId-of-callee, Text: "Hello World", ...}]
Success
Convert "Hello World" text-to-speech
Speak "Hello World"
ACTION_SUCCESSFUL(Speak)
Send next set of actions
Failure
Converting "Hello World" text-to-speech failed
ACTION_FAILED(Speak)
Send next set of actions
Caller
Callee
Amazon Chime SMA
Customer Lambda

In the diagram

A caller enters a number registered to a SIP media application, and the application responds as described for the previous diagram. When the Lambda function receives the `NEW_INBOUND_CALL` event, it returns the the section called "CallAndBridge" action to the SIP application. The application then uses the SIP `INVITE` method to send the `Trying (100)` and `Ringing (180)` responses to the callee.

If the callee answers, the SIP media application recieves a `200 (OK)` response, and it sends the same response to the caller. That establishes media, and the SIP application sends an `ACTION_SUCCESSFUL` event for the the section called "CallAndBridge" action to the Lambda function. The function then returns the Speak action and data to the SIP application, which converts

SpeakAndGetDigits

Play speech by providing text and gather dual tone multi-frequency (DTMF) digits from the user. The text can either be plain text or Speech Synthesis Markup Language (SSML)-enhanced text to provide more control over how the Amazon Chime SDK generates speech by adding pauses, emphasizing certain words, or changing the speaking style, among other supported SSML features. If a failure occurs, such as a user not entering the correct number of DTMF digits, the action plays the "failure" speech and then replays the main speech until the SIP media application exhausts the number of attempts defined in the `Repeat` parameter.

The Amazon Chime SDK uses Amazon Polly, a cloud service that converts text into lifelike speech, Amazon Polly provides both a standard and a neural engine for improved speech quality, more than 20 supported languages, and 60 voices. Amazon Polly provides speech features at no charge, but you do pay for using Amazon Polly. See the Amazon Polly pricing page or your billing dashboard for pricing information.

> **Important**
>
> Use of Amazon Polly is subject to the AWS Service Terms , including the terms specific to the AWS Machine Learning and Artificial Intelligence Services.

Topics

- Using the SpeakAndGetDigits action

- Handling ACTION_SUCCESSFUL events
- Handling ACTION_FAILED events
- Using the Amazon Chime SDK Voice Connector service-linked role

Using the SpeakAndGetDigits action

The following example shows a typical use of the `SpeakAndGetDigits` action:

```
{
    "SchemaVersion": "1.0",
    "Actions":[
        {
            "Type": "SpeakAndGetDigits",
            "Parameters": {
                "CallId": "call-id-1",          // required
                "InputDigitsRegex": "^\d{2}#$", // optional
                "SpeechParameters": {
                    "Text": "Hello World",      // required
                    "Engine": "neural",         // optional. Defaults to standard
                    "LanguageCode": "en-US",    // optional
                    "TextType": "text",         // optional
                    "VoiceId": "Joanna"         // optional. Defaults to Joanna
                },
                "FailureSpeechParameters": {
                    "Text": "Hello World",      // required
                    "Engine": "neural",         // optional. Defaults to the Engine
 value in SpeechParameters
                    "LanguageCode": "en-US",    // optional. Defaults to the
 LanguageCode value in SpeechParameters
                    "TextType": "text",         // optional. Defaults to the TextType
 value in SpeechParameters
                    "VoiceId": "Joanna"         // optional. Defaults to the VoiceId
 value in SpeechParameters
                },
                "MinNumberOfDigits": 3,         // optional
                "MaxNumberOfDigits": 5,         // optional
                "TerminatorDigits": ["#"],      // optional
                "InBetweenDigitsDurationInMilliseconds": 5000,  // optional
                "Repeat": 3,                    // optional
                "RepeatDurationInMilliseconds": 10000           // required
            }
        }
```

```
    ]
}
```

CallId

Description – The `CallId` of participant in the CallDetails of the Lambda function invocation.

Allowed values – A valid `callID`

Required – Yes

Default value – No

InputDigitsRegex

Description – A regular expression pattern to help ensure that users enter the correct digits and letters.

Allowed values – A valid regular expression pattern

Required – No

Default value – None

SpeechParameters.Engine

Description – Specifies the engine – standard or neural – to use when processing text for speech synthesis.

Allowed values – `standard` | `neural`

Required – No

Default value – Standard

SpeechParameters.LanguageCode

Description – Specifies the language code. This is only necessary if using a bilingual voice. If a bilingual voice is used and no language code is specified, the bilingual voice's default language is used.

Allowed values – Amazon Polly language codes

Required – No

Default value – None

SpeechParameters.Text

Description – Specifies the input text. If you specify `ssml` as the `SpeechParameters.TextType`, you must follow the SSML format for the input text. For more information about SSML, see Generating Speech from SSML Documents in the *Amazon Polly Developer Guide*.

Allowed values – String

Required – Yes

Default value – None

SpeechParameters.TextType

Description – Specifies the text format for `SpeechParameters.Text`. If not specified, `text` is used by default. For more information about SSML, see Generating Speech from SSML Documents in the *Amazon Polly Developer Guide*.

Allowed values – `ssml` | `text`

Required – No

Default value – `text`

SpeechParameters.VoiceId

Description – The ID of the Amazon Polly voice used to speak the text in `SpeechParameters.Text`.

Allowed values – Amazon Polly voice IDs

Required – No

Default value – Joanna

FailureSpeechParameters.Engine

Description – Specifies the engine – standard or neural – to use when processing the failure message used when the customer enters an invalid response for speech synthesis.

Allowed values – `standard` | `neural`

Required – No

Default value – The `SpeechParameters.Engine` value

FailureSpeechParameters.LanguageCode

Description – Specifies the language code used when the customer enters an invalid response. Only necessary when using a bilingual voice. If you use bilingual voice without specifying a language code, the bilingual voice's default language is used.

Allowed values – Amazon Polly language codes

Required – No

Default value – The `SpeechParameters.LanguageCode` value.

FailureSpeechParameters.Text

Description – Specifies the input text spoken when the customer enters an invalid response. If you specify `ssml` as the `FailureSpeechParameters.TextType`, you must follow the SSML format for the input text.

Allowed values – String

Required – Yes

Default value – None

FailureSpeechParameters.TextType

Description – Specifies whether the input text specified in `FailureSpeechParameters.Text` is plain text or SSML. The default value is plain text. For more information, see Generating Speech from SSML Documents in the *Amazon Polly Developer Guide*.

Allowed values – `ssml` | `text`

Required – No

Default value – The `SpeechParameters.Text` value

FailureSpeechParameters.VoiceId

Description – The ID for the voice used to speak the string in `FailureSpeechParameters.Text`.

Allowed values – Amazon Polly voice IDs

Required – Yes

Default value – The `SpeechParameters.VoiceId` value

MinNumberOfDigits

Description – The minimum number of digits to capture before timing out or playing the "call failed" message.

Allowed values – Greater than or equal to zero

Required – No

Default value – 0

MaxNumberOfDigits

Description – The maximum number of digits to capture before stopping without a terminating digit.

Allowed values – Greater than `MinNumberOfDigits`

Required – No

Default value – 128

TerminatorDigits

Description – Digit used to end input if the user enters less than the MaxNumberOfDigits

Allowed values – Any one of: 0 1 2 3 4 5 6 7 8 9 # or *

Required – No

Default value – #

InBetweenDigitsDurationInMilliseconds

Description – The wait time in milliseconds between digit inputs before playing the failure speech.

Allowed values – Greater than zero

Required – No

Default value – If not specified, defaults to the `RepeatDurationInMilliseconds` value

Repeat

Description – Total number of attempts to get digits. If you omit this parameter, the default is one attempt to collect digits.

Allowed values – Greater than zero

Required – No

Default value – 1

RepeatDurationInMilliseconds

Description – Timeout in milliseconds for each attempt to get digits.

Allowed values – Greater than zero

Required – Yes

Default value – None

Handling ACTION_SUCCESSFUL events

The following example shows a typical `ACTION_SUCCESSFUL` event.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 3,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
            "Type":  "SpeakAndGetDigits",
            "Parameters": {
                "CallId": "call-id-1",
                "InputDigitsRegex":  "^\d{2}#$",
                "SpeechParameters": {
                    "Engine":  "neural",
                    "LanguageCode": "en-US",
                    "Text":  "Hello World",
                    "TextType":  "text",
                    "VoiceId": "Joanna"
                },
                "FailureSpeechParameters": {
                    "Engine":  "neural",
```

```
                    "LanguageCode":  "en-US",
                    "Text":  "Hello World",
                    "TextType": "text",
                    "VoiceId": "Joanna"
                },
                "MinNumberOfDigits": 3,
                "MaxNumberOfDigits": 5,
                "TerminatorDigits": ["#"],
                "InBetweenDigitsDurationInMilliseconds": 5000,
                "Repeat": 3,
                "RepeatDurationInMilliseconds": 10000
            },
            "ReceivedDigits": "1234"
    },
    "CallDetails":{
       ...
    }
}
```

Handling ACTION_FAILED events

The following example shows a typical `ACTION_FAILED` event.

```
{
    "SchemaVersion": "1.0",
    "Sequence":2,
    "InvocationEventType": "ACTION_FAILED",
    "ActionData":{
            "Type":  "SpeakAndGetDigits",
            "Parameters": {
                "CallId": "call-id-1",
                "InputDigitsRegex":  "^\d{2}#$",
                "SpeechParameters": {
                    "Engine":  "neural",
                    "LanguageCode": "en-US",
                    "Text":  "Hello World",
                    "TextType":  "text",
                    "VoiceId": "Joanna"
                },
                "FailureSpeechParameters": {
                    "Engine":  "neural",
                    "LanguageCode":  "en-US",
                    "Text":  "Hello World",
                    "TextType": "text",
```

```
                    "VoiceId": "Joanna"
                },
                "MinNumberOfDigits": 3,
                "MaxNumberOfDigits": 5,
                "TerminatorDigits": ["#"],
                "InBetweenDigitsDurationInMilliseconds": 5000,
                "Repeat": 3,
                "RepeatDurationInMilliseconds": 10000
            },
            "ErrorType":  "SystemException",
            "ErrorMessage":  "System error while running action"
    },
    "CallDetails":{
       ...
    }
}
```

Error handling

This table lists and describes the error messages thrown by the the `Speak` action.

Error	Message	Reason
`AccessDenied`	The `AWSServiceRoleForAmazonChimeVoiceConnector` role is not configured correctly.	The role used to make requests to Amazon Polly doesn't exist or is missing permissions. To resolve, see the steps in the Using the Amazon Chime SDK Voice Connector service-linked role section
`InvalidActionParameter`		There was an error validating the action parameters. To review the available parameters for this action, and their options, see SynthesizeSpeech in the Amazon Polly Developer Guide.

Error	Message	Reason
`MissingRequiredActionParameter`	`Text` is a required parameter.	The action parameters must have a `Text` value
`MissingRequiredActionParameter`	`Text` is limited to 1,000 characters	The text exceeded the character limit.
`SystemException`	System error while running action.	A system error occurred while running the action.

Using the Amazon Chime SDK Voice Connector service-linked role

You don't need to manually create a service-linked role for the `Speak` or `SpeakAndGetDigits` actions. When you create or update a SIP media application in the Amazon Chime SDK console, the AWS Command Line Interface, or the AWS API, the Amazon Chime SDK creates the service-linked role for you.

For more information, see Using the Amazon Chime service-linked role in the *Amazon Chime SDK Administrator Guide*.

StartBotConversation

The `StartBotConversation` action establishes a voice conversation between an end user and your Amazon Lex v2 bot. The user provides the required information to the bot. The bot then returns the information to the public switched telephone network (PSTN) Audio Lambda function, and the function performs the requested tasks.

For example, the bot can play an optional welcome message at the start of a conversation to briefly describe the task that the PSTN Audio Lambda function can perform. The conversation goes back and forth between the user and the bot until the bot gathers the required information. Once the conversation ends, the Amazon Chime SDK invokes your PSTN Audio Lambda function with an action success event, which contains the information gathered by the bot. Your PSTN Audio Lambda function processes the information and performs the requested task.

The Audio Service provides life-like conversational interaction with your users. For example, users can interrupt the bot and answer a question before the audio prompt finishes. What's more, users can use any combination of voice and DTMF digits to provide information. The bot waits for the user to provide input before responding. You can configure how long the bot waits for the user to

finish speaking before interpreting any speech input. The user can also instruct the bot to wait if they need time to retrieve additional information during a call, such as credit card numbers.

The `StartBotConversation` action uses Amazon Lex and Amazon Polly for the duration of the bot conversation. Standard Amazon Lex and Amazon Polly costs apply. For more pricing information, see the Amazon Lex streaming conversation pricing, and Amazon Polly Pricing pages.

> **Note**
>
> You can't run this action on a bridged call, or on a call that has joined an Amazon Chime SDK meeting.

> **Important**
>
> Use of Amazon Lex and Amazon Polly is subject to the AWS Service Terms , including the terms specific to the AWS Machine Learning and Artificial Intelligence Services.

Topics

StartBotConversation syntax

The following example shows typical `StartBotConversation` syntax.

```
{
  "SchemaVersion": "1.0",
```

```
  "Actions":[
    {
      "Type": "StartBotConversation",
      "Parameters": {
        "CallId": "string",
        "ParticipantTag": "string",
        "BotAliasArn": "string",
        "LocaleId": "string",
        "Configuration": {
          "SessionState": {
             "SessionAttributes": {
                "string": "string"
             },
             "DialogAction" : {
               "Type": "string"
             }
          },
          "WelcomeMessages": [
            {
              "Content": "string",
              "ContentType": "string"
            }
          ]
        }
      }
    }
  ]
}
```

CallId

Description – The `CallID` of a participant in the `CallDetails` of the AWS Lambda function invocation. The `StartBotConversation` action uses this ID as the bot's `SessionId`. All bot conversations that take place on a call share the same conversation session. You can modify the session state between your user and your bot by using the Amazon Lex PutSession API. For more information, see Managing sessions with the Amazon Lex v2 API in the *Amazon Lex Developer Guide*.

Allowed values – A valid call ID.

Required – No, if `ParticipantTag` is present.

Default value – None.

ParticipantTag

Description – The `ParticipantTag` of one of the connected participants in the `CallDetails`.

Allowed values – `LEG-A`.

Required – No, if `CallId` is present.

Default value – `ParticipantTag` of the invoked `callLeg`. Ignored if you specify `CallDetails`.

BotAliasArn

Description – The bot alias ARN of your Lex bot. You must create the bot in the same AWS Region as your PSTN Audio application. A valid Amazon Lex bot alias has this format: `arn:aws:lex:`*`region`*`:`*`awsAccountId`*`:bot-alias/`*`botId`*`/`*`botAliasId`*, where *`region`* is the AWS Region where your bot resides. The *`awsAccountId`* is the AWS account ID in which your Amazon Lex bot is created. The `botId` value is the identifier assigned to the bot when you created it. You can find the bot ID in the Amazon Lex console on the **Bot details** page. The `botAliasId` is the identifier assigned to the bot alias when you created it. You can find the bot alias ID in the Amazon Lex console on the **Aliases** page.

Allowed values – A valid bot ARN.

Required –Yes.

Default value –None.

LocaleId

Description – The identifier of the locale that you used for your bot. For a list of locales and language codes, see Languages and locales supported by Amazon Lex.

Allowed values – Languages and locales supported by Amazon Lex.

Required – No.

Default value – en_US.

Configuration

Description – The conversation configuration, including session state and welcome messages. The total size of the JSON string representation of the `Configuration` object is limited to 10 KB.

Allowed values – `Configuration` object.

Required – No.

Default value – None.

Configuration.SessionState

Description – The state of the user's session with Amazon Lex v2.

Allowed values – `SessionState` object.

Required – No.

Default value – None.

Configuration.SessionState.SessionAttributes

Description – A map of the key/value pairs that represent session-specific context information. This map contains application information passed between Amazon Lex v2 and a client application.

Allowed values – String to string map.

Required – No.

Default value – None.

Configuration.SessionState.DialogAction.Type

Description – The next action that the bot takes in its interactions with the user. Possible values:

- *Delegate* Amazon Lex v2 determines the next action.
- *ElicitIntent* The next action elicits an intent from the user.

Allowed values – `Delegate` | `ElicitIntent`.

Required – No.

Default value – None.

Configuration.WelcomeMessages

Description – A list of messages to send to the user at the start of the conversation. If you set the `welcomeMessage` field, you must set the `DialogAction.Type` value to `ElicitIntent`.

Allowed values – Message object

Required – No.

Default value – None.

Configuration.WelcomeMessages.Content

Description – The welcome message text.

Allowed values – String.

Required – No.

Default value – None.

Configuration.WelcomeMessages.ContentType

Description – Indicates the welcome message type.

Allowed values – `PlainText` | `SSML`

- *PlainText* – The message contains plain UTF-8 text.
- *SSML* – The message contains text formatted for voice output.

Required – Yes.

Default value – None.

Using the StartBotConversation action

The following example shows a typical `StartBotConversation` action.

```
{
  "SchemaVersion": "1.0",
  "Actions":[
    {
      "Type": "StartBotConversation",
      "Parameters": {
        "CallId": "call-id-1",
        "BotAliasArn": "arn:aws:lex:us-east-1:123456789012:bot-
alias/ABCDEFGHIH/MNOPQRSTUV",
        "LocaleId": "en_US",
        "Configuration": {
          "SessionState": {
             "SessionAttributes": {
                "mykey1": "myvalue1"
             },
```

```
              "DialogAction" : {
                "Type": "ElicitIntent"
              }
          },
          "WelcomeMessages": [
            {
              "Content": "Welcome. How can I help you?",
              "ContentType": "PlainText"
            }
          ]
        }
      }
    }
  ]
}
```

Handling ACTION_SUCCESSFUL events

The following example shows a typical `ACTION_SUCCESSFUL` event for the `StartBotConversation` action.

```
{
    "SchemaVersion": "1.0",
    "Sequence": number,
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData":
    {
        "CallId": "string",
        "Type": "StartBotConversation",
        "Parameters": {
            // parameters provided in the StartBotConversation action.
        },
        "CallDetails": {
            // Information about the call associated with the AWS Lambda invocation.
        },
        "IntentResult": {
            "SessionId": "string",
            "SessionState": {
                "SessionAttributes": {
                    "string": "string"
                },
                "Intent": {
                    "Name": "string",
```

```
                    "Slots":  {
                        "string": {
                            "Value": {
                                "OriginalValue": "string",
                                "InterpretedValue": "string",
                                "ResolvedValues": ["string"]
                            },
                            "Values": []
                        }
                    },
                    "State": "string",
                    "ConfirmationState": "string"
                }
            },
            "Interpretations": [
                {
                    "NluConfidence": {
                        "Score": number
                    },
                    "Intent": {
                        "Name": "string",
                        "Slots": {
                            "string": {
                                "Value": {
                                    "OriginalValue": "string",
                                    "InterpretedValue": "string",
                                    "ResolvedValues": ["string"]
                                },
                                "Values": []
                            }
                        },
                        "State": "string",
                        "ConfirmationState": "string"
                    }
                }
            ]
        }
    }
}
```

IntentResult

The result of the conversation between the user and the bot.

SessionId

The identifier of the bot conversation session. When a user starts a conversation with your bot, Amazon Lex creates a session. A session encapsulates the information exchanged between your user and the bot. The `StartBotConversation` action uses the call ID as the bot's `SessionId`. You can modify the session state between your user and your bot by using the Lex PutSession API. For more information, see Managing sessions with the Amazon Lex V2 API in the *Amazon Lex Developer Guide*.

SessionState

The state of the user's Amazon Lex v2 session.

SessionState.SessionAttributes

Map of key/value pairs that represent session-specific context information. The map contains bot conversation information passed between the Lambda function attached to your bot and the PSTN Audio Lambda function.

Interpretations

A list of intents derived by Amazon Lex that may satisfy the a customer's utterance. The intent with the highest `NluConfidence` score becomes the Intent for the `SessionState`.

Interpretations.NluConfidence.Score

A score that indicates how confident Amazon Lex v2 is that an intent satisfies a user's intent. Ranges between 0.00 and 1.00. Higher scores indicate higher confidence.

Intent

The action the user wants to perform.

Intent.Name

The name of the intent.

Intent.Slots

A map of all of the slots for the intent. The name of the slot maps to the value of the slot. If a slot has not been filled, the value is null.

Intent.Slots.Value

The value of the slot.

Intent.Slots.Values

A list of one or more values that the user provided for the slot.

Intent.Slots.Value.OriginalValue

The text of the user's reply, entered for the slot.

Intent.Slots.Value.InterpretedValue

Description – The value that Amazon Lex v2 determines for the slot. The actual value depends on the bot's value selection strategy setting. You can choose to use the value entered by the user, or you can have Amazon Lex v2 choose the first value in the `resolvedValues` list.

Intent.Slots.Value.ResolvedValues

A list of additional values that Amazon Lex v2 recognizes for the slot.

Intent.State

Description – Fulfillment information for the intent. Possible values:

- `Failed` – The Lambda function failed to fulfill the intent.
- `Fulfilled` – The Lambda function fulfilled the intent.
- `ReadyForFulfillment` – The information for the intent is present, and your Lambdafunction can fulfill the intent.

Intent.ConfirmationState

Description – Indicates confirmation of the intent. Possible values:

- *Confirmed* – The Intent is fulfilled.
- *Denied* – The user responded "no" to the confirmation prompt.
- *None* – The user wasn't prompted for confirmation, or the user was prompted but didn't confirm or deny the prompt.

Handling ACTION_FAILED events

The following example shows a typical `ACTION_FAILED` event for the `StartBotConversation` action.

```
{
```

```
    "SchemaVersion": "1.0",
    "Sequence": number,
    "InvocationEventType": "ACTION_FAILED",
    "ActionData":{
        "CallId": "string",
        "Type": "StartBotConversation",
        "Parameters": {
            // parameters provided in the StartBotConversation action
        },
        "ErrorType": "string",
        "ErrorMessage": "string"
    },
    "CallDetails":{
    }
}
```

ErrorType

A string that uniquely identifies an error condition.

ErrorMessage

A generic description of the error condition.

Error codes

The following table lists the error messages that a Lambda function can return in an `ACTION_FAILED` event.

Error	Description
`InvalidActionParameter`	One or more action parameters are invalid. The error message describes the invalid parameter.
`SystemException`	A system error occurred while running an action.
`ResourceNotFound`	A specified bot is not found.
`ResourceAccessDenied`	Access to the bot is denied.

Error	Description
`ActionExecutionThrottled`	Bot conversation service limit is exceeded. The error message describes the specific service limit exceeded.

Granting permissions to use a bot

The following example grants the Amazon Chime SDK permission to call the Amazon Lex StartConversation APIs. You must explicitly grant the Audio Service permission to use your bot. The condition block is required for service principals. The condition block must use the global context keys `AWS:SourceAccount` and `AWS:SourceArn`. The `AWS:SourceAccount` is your AWS account ID. The `AWS:SourceArn` is the resource ARN of the PSTN Audio application that invokes the Lex bot.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowChimePstnAudioUseBot",
      "Effect": "Allow",
      "Principal": {
        "Service": "voiceconnector.chime.amazonaws.com"
      },
      "Action": "lex:StartConversation",
      "Resource": "arn:aws:lex:region:awsAccountId:bot-alias/botId/aliasId",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "awsAccountId"
        },
        "ArnEquals": {
        "AWS:SourceArn": "arn:aws:voiceconnector:region:awsAccountId:sma/smaId"
        }
      }
    }
  ]
}
```

Configuring voice and DTMF timeouts

You can configure the voice and DTMF timeouts when capturing user input. You can configure timeouts through session attributes when starting a conversation with a bot, and overwrite them in your Lex bot's Lambda function if necessary. Amazon Lex lets you set multiple slots for an intent or bots. Because you can specify that session attributes apply to the intent and slot level, you can specify that the attribute is set only when you're collecting a certain type of input. For example, you can specify a longer time-out when you're collecting an account number than when you're collecting a date. You can use wildcards in the session attribute key.

For example, to set a voice timeout for all slots for all intents to 4000 milliseconds, you can provide a session attribute using: `x-amz-lex:start-timeout-ms:*:*` as the session attribute name and `4000` as the session attribute value. For more information, see Configuring timeouts for capturing user input in the *Amazon Lex Developer Guide*.

Using DTMF inputs during a conversation

Amazon Lex bots support voice and keypad input during a conversation. The bots interpret keypad input as DTMF digits. You can prompt contacts to end their input with a pound key (#) and to cancel a conversation by using the star key (*). If you don't prompt customers to end their input with the pound key, Lex stops waiting for additional key presses after 5 seconds.

Billing and service quotas

AWS bills you for the following costs:

- Amazon Chime SDK usage for the call. For more information, see Amazon Chime SDK pricing.
- Amazon Lex usage for interpreting users' speech. For more information, see Amazon Lex streaming conversation pricing.
- Amazon Polly usage for synthesizing text responses from your bot. For more information, see Amazon Polly Pricing.

You also need to be aware of the following service quotas:

- The Amazon Chime SDK has a service quota for the maximum number of Amazon Lex bots you can use with the PSTN Audio StartBotConversation action. For more information, refer to SIP trunking and voice quotas, in the *AWS General Reference*.

- Amazon Lex has a service quota for the maximum number of concurrent voice conversations per Lex bot. You can contact the Amazon Lex service team for quota increases. For more information, see the Amazon Lex Guidelines and quotas in the *Amazon Lex Developer Guide*.
- Amazon Polly has a service quota for synthesizing text responses. You can contact the Amazon Polly service team for quota increases. For more information about Amazon Polly service quotas, see Quotas in Amazon Polly, in the *Amazon Polly Developer Guide*.

Using SIP headers

You can now send and receive a User-To-User header, a Diversion header, and custom SIP headers in your AWS Lambda functions when you want to exchange call context information with your SIP infrastructure.

- The User-to-User (UUI) header can be used to send call control data. This data is inserted by the application initiating a session and used by the application accepting the session. It is not used for any basic SIP functionality. For example, you could use the UUI header in a call center to pass information between agents about a call.
- The Diversion header is used to show from where the call was diverted and why. You can use this header to either see diversion information from other SIP agents, or pass it along.
- Custom SIP Headers allow you to pass along any other information you want. For example, if you want to pass along an account id, you can create an X header called “X-Account-Id” and add this information.

You must prefix your custom SIP headers with `x-`. The headers are exposed in the AWS Lambda function and received as part of a `NEW_INBOUND_CALL` event during an inbound call. You can also include these headers in outbound call legs when triggering a CallAndBridge action or the CreateSipMediaApplicationCall API.

The `Participants` section of a Lambda function contains the `SipHeaders` field. This field is available when you receive a custom header, or when you populate the `User-to-User` or `Diversion` header.

This example shows an expected response when an AWS Lambda invocation contains SIP headers.

```
{
    "SchemaVersion": "1.0",
    "Sequence": 3,
```

```
    "InvocationEventType": "ACTION_SUCCESSFUL",
    "ActionData": {
        "Type":"actionType",
        "Parameters":{
            // Parameters vary by actionType
        }
    },
    "CallDetails":{
        .....
        .....
        "Participants":[
            {
                "CallId": "call-id-1",
                "ParticipantTag": "LEG-A",
                .....
                "Status": "Connected"
                "SipHeaders": {
                    "X-Test-Value": "String",
                    "User-to-User":
 "616d617a6f6e5f6368696d655f636f6e6e6563745f696e746567726174696f6e;encoding=hex",
                    "Diversion": "sip:
+11234567891@public.test.com;reason=unconditional"
                }
            },
            {
            "CallId": "call-id-2",
            "ParticipantTag": "LEG-B",
            .....
            "Status": "Connected"
            }
        ]
    }
}
```

The following example shows a successful CallAndBridge action, due to an invalid entry for the `SipHeaders` parameter.

```
{
    "SchemaVersion": "1.0",
    "Actions":[
        {
            "Type": "CallAndBridge",
            "Parameters":{
```

```
            "CallTimeoutSeconds": 30,
            "CallerIdNumber": "e164PhoneNumber", // required
            "RingbackTone": { // optional
                "Type": "S3",
                "BucketName": "s3_bucket_name",
                "Key": "audio_file_name"
            },
            "Endpoints":[
               {
                    "Uri":"e164PhoneNumber", // required
                    "BridgeEndpointType":"PSTN" // required
               }
            ],
            "SipHeaders": {
                "X-Test-Value": "String",
                "User-to-User":
 "616d617a6f6e5f6368696d655f636f6e6e6563745f696e746567726174696f6e;encoding=hex",
                "Diversion": "sip:+11234567891@public.test.com;reason=unconditional"
            }
         }
      }
   ]
}
```

The following example shows a failed CallAndBridge action caused by an invalid `SipHeaders` parameter.

```
{
    "SchemaVersion":"1.0",
    "Sequence":3,
    "InvocationEventType":"ACTION_FAILED",
    "ActionData":{
        "Type":"actionType",
        "Parameters":{
            // Parameters vary by Action Type
            "SipHeaders": {
                "X-AMZN": "String",
                "User-to-User":
 "616d617a6f6e5f6368696d655f636f6e6e6563745f696e746567726174696f6e;encoding=hex",
                "Diversion": "sip:+11234567891@public.test.com;reason=unconditional"
             },
        },
        "ErrorType": "InvalidActionParameter",
```

```
        "ErrorMessage": "Invalid SIP header(s) provided: X-AMZN"
    },
    "CallDetails":{
       .....
       "Participants":[
          {
             "CallId":"call-id-1",
             "ParticipantTag":"LEG-A",
             .....
             "Status":"Connected"
          },
          {
             "CallId":"call-id-2",
             "ParticipantTag":"LEG-B",
             .....
             "Status":"Connected"
          }
       ]
    }
}
```

Using the SipHeaders field

When you trigger the CreateSipMediaApplicationCall API, the optional `SipHeaders` field allows you to pass custom SIP headers to an outbound call leg. Valid header keys must include one of the following:

- The `x-` prefix
- The `User-to-User` header
- The `Diversion` header

`X-AMZN` is a reserved header. If you use this header in an API call, it will fail. The headers can be a maximum length of 2048 characters.

The following example shows a typical CreateSipMediaApplicationCall API in the command-line interface with the optional `SipHeaders` parameter.

```
create-sip-media-application-call
    --from-phone-number value // (string)
    --to-phone-number value // (string)
```

```
    --sip-media-application-id value // (string)
    --sip-headers // (map)
```

For more information, see A Mechanism for Transporting User-to-User Call Control Information in SIP and Diversion Indication in SIP.

Using call detail records

Amazon Chime SDK administrators can configure Amazon Chime SDK Voice Connectors to store *call detail records* (CDRs). For more information about configuring Amazon Chime SDK Voice Connectors to store CDRs, see Managing global settings in Amazon Chime SDK in the *Amazon Chime SDK Administration Guide*.

Once you enable CDRs, after each call the SIP media application sends the records to a folder named **Amazon-Chime-SMADRs** in your S3 bucket.

The following table lists the attributes of a CDR, and shows their proper formatting. The records contain all the fields listed here for all calls.

Value	Description
`"AwsAccountId":" AWS-account-ID "`	The AWS account ID associated with the SIP media application that initiated the PSTN usage
`"TransactionId":" transaction-ID "`	The transaction ID of the call
`"CallId":" SIP-media-application-call-ID "`	The call ID of the participant for the associate d usage
`"VoiceConnectorId":" voice-connector-ID "`	Amazon Chime SDK Voice Connector ID UUID
`"Status":" status"`	Status of the call (Completed, Failed)
`"BillableDurationSeconds":" billable-duration-in-seconds "`	Billable duration of the call in seconds
`"SchemaVersion":" schema-version "`	The CDR schema version

Value	Description
"SourcePhoneNumber":" *12075550155* "	E.164 origination phone number
"SourcePhoneNumberName":" *North Campus Reception* "	The name assigned to the source phone number
"DestinationPhoneNumber":" *13605551214* "	E.164 destination phone number
"DestinationPhoneNumberName":" *South Campus Reception* "	The name assigned to the destination phone number
"UsageType":" *usage-type* "	Usage details of the line item in the Price List API
"ServiceCode":" *service-code* "	The code of the service in the Price List API
"Direction":" *direction* "	Direction of the call, `Outbound` or `Inbound`
"TimeStampEpochSeconds":" *start-time-epochseconds* "	The timestamp of the record in epoch/Unix timestamp format
"Region":" *AWS-region* "	AWS Region for the Amazon Chime SDK Voice Connector
"SipRuleId":" *sip-rule-id* "	The ID of the sip rule that is triggered when a call reaches the PSTN Audio service
"SipApplicationId":" *sip-application-id* "	The ID of the SIP application that handles a call
"CallLegTriggerType":" *trigger-type* "	The type of event that triggered a call
"BillableVoiceFocusSeconds":" *billable-voice-focus-in-seconds* "	The billable amount of Voice Focus usage, in seconds

Understanding timeouts and retries

The PSTN Audio service interacts with AWS Lambda functions synchronously. Applications wait 5 seconds for AWS Lambda functions to respond before retrying an invocation. When a function returns an error with one of the *4XX* status codes, then by default the SIP media application only retries the invocation once. If you run out of retries, calls terminate with the `480 Unavailable` error code. For more information about AWS Lambda errors, see Troubleshoot invocation issues in AWS Lambda.

Debugging and troubleshooting

Use the following information to help you diagnose and fix common issues that you might encounter when working with the Amazon Chime SDK PSTN Audio service.

Topics

Checking the logs

If you're debugging a SIP media application, check the Cloudwatch logs for the AWS Lambda function associated with the application.

Next, check the logs associated with the SIP media application. As needed, you can configure the SIP media application for logging. For more information, see Using SIP media applications in the *Amazon Chime SDK Administrator Guide*. If you enable logging, you can find the logs on Cloudwatch, in the /aws/ChimeSipMediaApplicationSipMessages/ *SIP media application Id* log group.

Debugging unexpected hangups

- Verify that your AWS Lambda policy grants the `lambda:InvokeFunction` permission to the voiceconnector.chime.amazonaws.com service principal.
- Check the logs for your AWS Lambda function to ensure that it's being successfully invoked.
- If the logs show incoming events and returned actions, verify that you don't return a hangup action in when the AWS Lambda function is invoked.

- Check the Cloudwatch logs for your SIP media application. The following table lists some of the messages you may encounter.

Message	Resolution
AWS Lambda client operation timed out.	The function took longer than 20 seconds to complete. Shorten the response time to less than 20 seconds.
Access denied while invoking the AWS Lambda function.	The AWS Lambda function doesn't provide a policy that allows the service to access the Amazon Chime SDK Voice Connector service principal. Provide the `voiceconnector.chime.amazonaws.com` service principal with the `lambda:InvokeFunction` permission in your AWS Lambda policies.
The AWS Lambda function was throttled.	The Audio Service couldn't call your AWS Lambda function because the function was throttled. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/lambda-troubleshoot-throttling/.
Error while reading actions list.	The PSTN Audio service failed to parse the actions returned by your AWS Lambda function. Check the logs for `ACTION_FAILED` events, and consult the documentation for the failed action to ensure that you coded it properly.
The schema version in the invocation request does not match the schema version in the response.	Check your logs and ensure that your request and response use the same schema version.

Message	Resolution
Unsupported action *name* specified	The AWS Lambda function returned an action that the PSTN Audio service didn't recognize. Ensure the action is spelled correctly, and see the documentation for the action.
Actions list is empty.	The response to a `NEW_INCOMING_CALL` event didn't return any actions. Return an action in response to that event.
Too many actions specified in a response.	You returned more than 10 actions in response to an AWS Lambda invocation. Return 10 or fewer actions.
Response is blank or empty	You returned a null or an empty string. Make sure the response object includes at least the `SchemaVersion` field.

Debugging unexpected ACTION_FAILED events

If you receive an unexpected `ACTION_FAILED` event, check the following:

Actions	Error Type	Error Message	Resolution
CallAndBridge, PlayAudio, and PlayAudioAndGetDigits	`InvalidAudioSource`	Cannot access S3 bucket or audio file.	• Ensure the S3 bucket is in the same AWS account as the SIP media application. • Ensure the S3 bucket has given the `s3:GetObject` permissions to the `voiceconnector.chime.amazon`

Actions	Error Type	Error Message	Resolution
			`aws.com` service principal.
PlayAudio, and PlayAudioAndGetDigits	`InvalidAudioSource`	Audio Source parameter value is invalid.	• Ensure you use a valid `Type`, such as `S3`. • Ensure that the S3 bucket grants the `s3:GetObject` permissions to the `voiceconnector.chime.amazonaws.com` service principal. • Ensure the `BucketName` field is not null or empty. • Ensure the `Key` field is not null or empty.
CallAndBridge	`InvalidAudioSource`	Ringtone parameter value is invalid.	• Ensure you use a valid `Type`, such as `S3`. • Ensure the `BucketName` field is not null or empty. • Ensure the `Key` field is not null or empty.
	`InvalidActionParameter`	Invalid number of endpoints provided.	Ensure the Endpoints are not null or zero and not greater than one.

Actions	Error Type	Error Message	Resolution
	`InvalidActionParameter`	Endpoint parameter is invalid.	• Ensure Endpoint URI value is provided. • If the Endpoint Type is PSTN, make sure the phone number provided in the Uri field is a valid E.164 phone number. • If the Endpoint Type is PSTN, make sure that the ARN field is not set or is set to null.
	`InvalidActionParameter`	Invalid caller ID.	Provide a valid E.164 formatted phone number in the `CallerId` field.
	`InvalidActionParameter`	Caller ID not defined.	Provide a valid E.164 formatted phone number in the `CallerId` field.
	`InvalidActionParameter`	The `MaxCallTimeout` parameter is invalid. Timeout must be between 0 and 120 seconds.	Set the `MaxCallTimeout` interval to a value between 0 and 120 seconds.

Actions	Error Type	Error Message	Resolution
	`InvalidActionParameter`	Provided caller ID number is invalid. Number must be owned by this AWS account, or be the `From` number of LEG-A.	• Ensure the `CallerId` number is provisioned and associated with the same AWS account as the SIP media application. • If the number is not associated with the account, it must match the number in the `From` field of LEG-A.
	`InvalidActionParameter` .	Invalid SIP header(s) provided: {Header}.	• Remove any internal custom headers: `x-vine`, `x-amzn`, `x-vc`, `x-canary`, `x-voice`. • Ensure your custom headers begin with `x-`. You can also set them to `user-to-user` or `diversion` .

Actions	Error Type	Error Message	Resolution
JoinChimeMeeting	`InvalidActionParameter` .	JoinToken parameter value is invalid.	• Verify that the meeting join token is correct. • Verify that the participant that the token is associated with is still a valid attendee of the meeting. • Verify that the meeting still exists.
ModifyChimeMeetingAttendee (muting and unmuting audio)	`InvalidActionParameter`	The value in the `Operation` field of the `ModifyChimeMeetingAttendees` action is invalid.	Ensure that the service supports muting and unmuting operations.
	`InvalidActionParameter`	The meeting ID parameter is invalid.	Ensure the meeting ID is correct.
	`InvalidActionParameter`	Attendee List parameter is invalid.	You provided no attendees, or you provided more than 100 attendees. Provide between 1 and 100 attendees.

Actions	Error Type	Error Message	Resolution
	`InvalidActionParameter`	One or more attendees are not part of this meeting. All attendees must be part of this meeting.	One of the attendees provided in the action is not a valid participant of the meeting specified. Remove any attendee not in the meeting.
Pause	`InvalidActionParameter`	The `Duration` parameter is invalid.	Set the pause duration to between 100 and 30000.
PlayAudioAndGetDigits	`InvalidActionParameter`	The `MaxNumberOfDigits` parameter is invalid.	Ensure `MaxNumberOfDigits` is between 0 and 128, and that it's greater than `MinNumberOfDigits` /
	`InvalidActionParameter`	The `RepeatDurationInMilliseconds` parameter is invalid.	Ensure the `RepeatDurationInMilliseconds` value is positive.
	`InvalidActionParameter`	The `InputDigitsRegex` parameter is invalid.	Ensure the `InputDigitsRegex` is a valid regex pattern.
ReceiveDigits	`InvalidActionParameter`	The `InBetweenDigitsDurationInMilliseconds` parameter is invalid.	Ensure the value is greater than 0.

Actions	Error Type	Error Message	Resolution
	InvalidActionParameter	The FlushDigitsDurationInMilliseconds parameter is invalid.	The FlushDigitsDurationInMilliseconds interval is less than or equal to the InBetweenDigitsDurationInMilliseconds interval. Make the InBetweenDigitsDurationInMilliseconds interval greater than the FlushDigitsDurationInMilliseconds interval.
	InvalidActionParameter	The InputDigitsRegex parameter is invalid.	Ensure the value is not empty or null.

Actions	Error Type	Error Message	Resolution
RecordAudio	InvalidActionParameter	The RecordingDestination parameter is invalid.	• Verify that the Type field is valid, such as S3. • Verify that the BucketName field is not empty or null. • Verify that the prefix consists of valid characters. • Verify that the prefix is less than or equal to 979 bytes.
	InvalidActionParameter	The DurationInSeconds parameter is invalid.	DurationInSeconds must not be null and must be greater than 0.
	InvalidActionParameter	The SilenceThreshold parameter is invalid.	SilenceThreshold must not be null and must be between 1 and 1000.
	InvalidActionParameter	The SilenceDurationInSeconds parameter is invalid.	SilenceDurationInSeconds must not be null and must be greater than 0.

Actions	Error Type	Error Message	Resolution
	`InvalidActionParameter`	An error occurred while uploading the recording to the S3 bucket.	• Ensure the S3 bucket is in the same AWS account as the SIP media application. • Ensure that the S3 bucket has granted `s:PutObject` and `s:PutObjectAcl` permissions to use to the `voiceconnector.chime.amazonaws.com` service principal.
Understanding VoiceFocus	`MissingRequiredActionParameter`	Missing a required action parameter.	Provide a valid boolean value for the `Enable` parameter.

Understanding VoiceFocus

Enables you to apply Amazon Voice Focus noise suppression to inbound and outbound call legs on a public switched telephony network (PSTN) call. When you apply Amazon Voice Focus, it reduces background noise without impacting human speech. This can make the current speaker easier to hear.

To create inbound call legs, you use a SIP rule that invokes an AWS Lambda function with a `NewInboundCall` event. You can create outbound call legs by using the CallAndBridge action, or by using a CreateSIPMediaApplicationCall API operation. For more information about Amazon Voice Focus, see How the Amazon Chime SDK's noise cancellation works.

Amazon Voice Focus reduces unwanted, non-speech noises, including:

- **Environment noises**—wind, fans, running water

- **Background noises**—lawnmowers, barking dogs
- **Foreground noises**—typing, paper shuffling

> **Note**
>
> When you use Amazon Voice Focus, AWS bills you for the active call minutes of each call leg and for each minute of SIP media application usage.

This example shows a typical `VoiceFocus` action.

```
{
    "SchemaVersion": "1.0",
    "Actions":[
        {
            "Type": "VoiceFocus",
            "Parameters": {
                "Enable": True|False,            // required
                "CallId": "call-id-1",           // required
            }
        }
    ]
}
```

Enable

Description – Enables or disables Amazon Voice Focus

Allowed values – `True` | `False`

Required – Yes

Default value – None

CallId

Description – CallId of participant in the `CallDetails` of the AWS Lambda function invocation

Allowed values – A valid call ID

Required – Yes

Default value – None

This example shows a successful `ACTION_SUCCESSFUL` event for the `VoiceFocus` action.

```
{
   "SchemaVersion": "1.0",
   "Sequence": 3,
   "InvocationEventType": "ACTION_SUCCESSFUL",
   "ActionData": {
      "Type": "VoiceFocus",
      "Parameters": {
         "Enable": True,
         "CallId": "call-id-1"
      }
   },
   "CallDetails":{
      .....
      .....
      "Participants":[
         {
            "CallId": "call-id-of-caller",
            .....
            "Status": "Connected"
         },
         {
            "CallId": "call-id-of-callee",
            .....
            "Status": "Connected"
         }
      ]
   }
}
```

This example shows a typical `ACTION_FAILED` event for the `VoiceFocus` action.

```
{
   "SchemaVersion": "1.0",
   "Sequence":2,
   "InvocationEventType": "ACTION_FAILED",
      "ActionData":{
      "Type": "VoiceFocus",
      "Parameters": {
         "Enable": True,
         "CallId": "call-id-1"
      }
```

```
      },
      "ErrorType": "SystemException",
      "ErrorMessage": "System error while running action"
   },
   "CallDetails":{
      .....
      .....
      "Participants":[
         {
            "CallId": "call-id-of-caller",
            .....
         }
      ]
   }
}
```

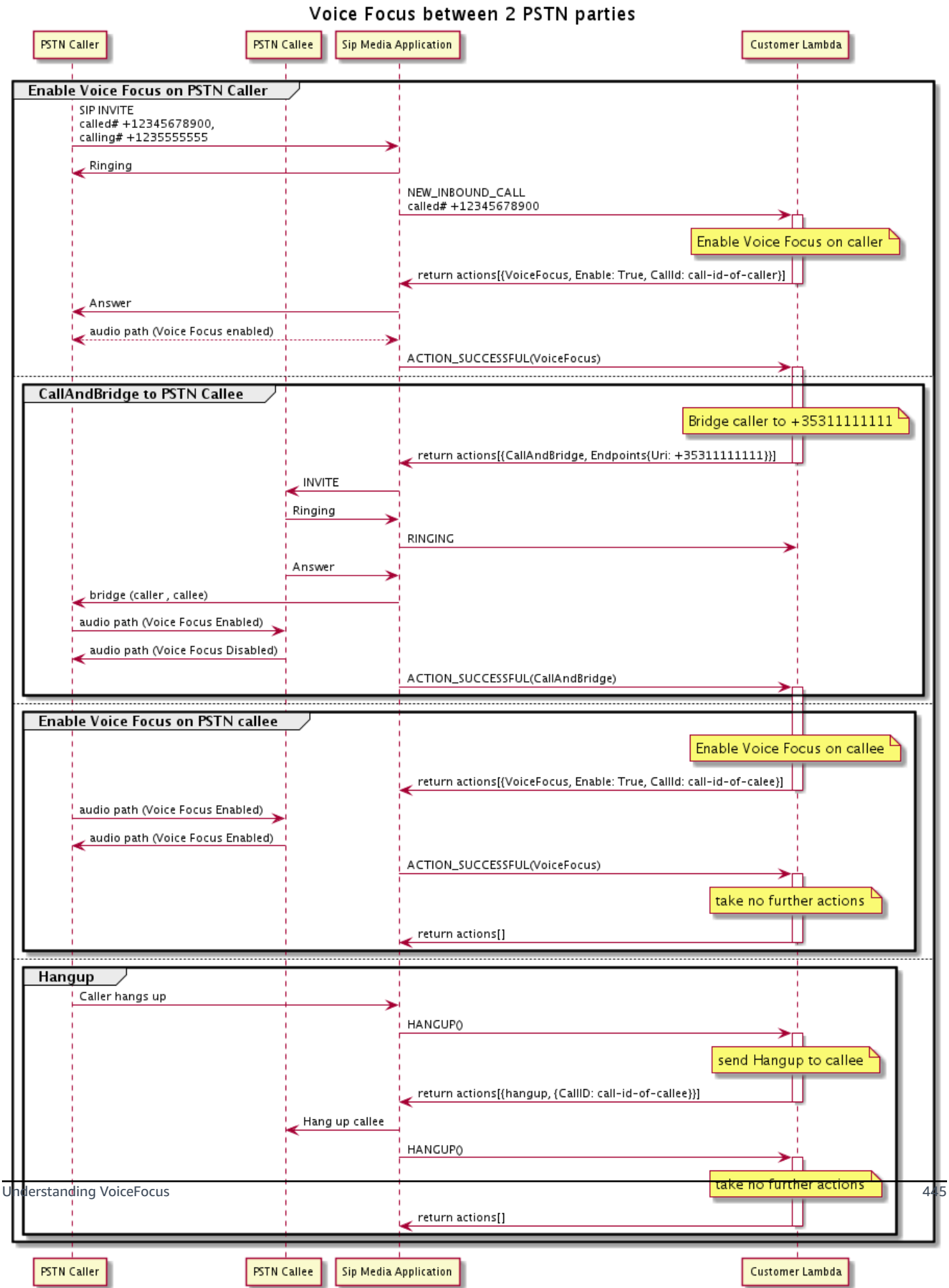
Error handling

For security reasons, the PSTN Audio actions have a limit of 5 call requests per second, per customer account (CPS). When call requests exceed the 5 CPS limit, the action returns an error message. This table lists the error messages returned by the `VoiceFocus` action.

Error	Message	Reason
`ActionExecutionThrottled`	Failed to run action. The maximum number of actions per second has been reached.	The number of Voice Focus action requests per second exceeded the system limit.
`MissingRequiredActionParameter`	Missing required action parameter.	Missing one or more of the required parameters while running the action.
`SystemException`	System error while running action.	A system error occurred while running the action.

Call flows

This diagram shows the call flow for enabling and disabling Amazon Voice Focus for a `CallAndBridge` action between two PSTN calls.


Voice Focus between 2 PSTN parties
PSTN Caller
PSTN Callee
Sip Media Application
Customer Lambda
Enable Voice Focus on PSTN Caller
SIP INVITE
called# +12345678900,
calling# +1235555555
Ringing
NEW_INBOUND_CALL
called# +12345678900
Enable Voice Focus on caller
return actions[{VoiceFocus, Enable: True, CallId: call-id-of-caller}]
Answer
audio path (Voice Focus enabled)
ACTION_SUCCESSFUL(VoiceFocus)
CallAndBridge to PSTN Callee
Bridge caller to +35311111111
return actions[{CallAndBridge, Endpoints{Uri: +35311111111}}]
INVITE
Ringing
RINGING
Answer
bridge (caller , callee)
audio path (Voice Focus Enabled)
audio path (Voice Focus Disabled)
ACTION_SUCCESSFUL(CallAndBridge)
Enable Voice Focus on PSTN callee
Enable Voice Focus on callee
return actions[{VoiceFocus, Enable: True, CallId: call-id-of-calee}]
audio path (Voice Focus Enabled)
audio path (Voice Focus Enabled)
ACTION_SUCCESSFUL(VoiceFocus)
take no further actions
return actions[]
Hangup
Caller hangs up
HANGUP()
send Hangup to callee
return actions[{hangup, {CallID: call-id-of-callee}}]
Hang up callee
HANGUP()
take no further actions
return actions[]
PSTN Caller
PSTN Callee
Sip Media Application
Customer Lambda

For the outbound call leg, the AWS Lambda function enables Amazon Voice focus for the caller and returns a set of actions, including `CallAndBridge`. Once the call is bridged, the `VoiceFocus` action returns an `ACTION_SUCCESSFUL` event, and the Lambda function returns another set of events that enables Amazon Voice Focus for the person being called. That set of actions includes `VoiceFocus`, `Enable`, `True`, and the caller's ID. No further action is taken until the caller hangs up. The Lambda function then sends a `Hangup` action to the SIP media application. The application hangs up the person being called and sends a Hangup function back to the Lambda function, which takes no further actions.

PSTN audio service glossary

A

Action

In an AWS Lambda function, an action is an item that you want to run on a leg of a phone call, such as sending or receiving digits, joining a meeting, and so on. For more information about actions supported by the PSTN Audio Service, see Supported actions for the PSTN Audio service.

AWS Lambda

A compute service that lets you run code for almost any type of application or backend service without provisioning or managing servers.

AWS Lambda function

In the context of the PSTN Audio service, a function run in response to data passed by a SIP media application, such as placing an outbound call.

C

Call detail record

Data from Amazon Chime SDK Voice Connector calls, such as account IDs, source phone numbers, and destination countries. The records land as objects in an Amazon Simple Storage Service (S3) bucket in your account. For more information, see Managing global settings in Amazon Chime SDK in the *Amazon Chime SDK Administrator Guide*. For information about the record schema, see Using call detail records in this guide.

call ID

The ID assigned to the legs of all incoming calls.

Call leg

A part of a call. In Amazon Chime SDK applications, calls can come from valid phone numbers, a PSTN, or Amazon Chime SDK Voice Connectors. For more information, see Learn about using PSTN Audio service call legs in this guide.

Carrier

A company that provides mobile services. Short for **wireless carrier**.

Amazon Chime

A unified communications and collaboration service provided by AWS.

Amazon Chime SDK

A software development kit used by developers to add real-time media and communications to custom communication applications.

E

E.164

The only accepted format for phone numbers in the PSTN Audio service. An ITU-T recommendation, numbers use a 1-3 digit country code, followed by a maximum 12-digit subscriber number. For example: US: `+14155552671`, UK: `+442071838750 44`, Australia: `+61285993444`.

Endpoint

A hardware device or software service, such as a phone or a unified communications application.

EventBridge

A serverless event bus service that allows you to connect your applications to data from a variety of sources.

> **Note**
>
> SIP media applications do not send data to EventBridge. For more information, see Automating the Amazon Chime SDK with EventBridge in the *Amazon Chime SDK Administrator Guide*.

I

IVR

Interactive Voice Response. A system that allows people to interact with a computer-operated phone system by voice recognition or touchtone keypads.

L

Leg

See Call leg.

M

Media

The audio, video, or chat messages available for use during an Amazon Chime SDK meeting. A custom communications application can contain one or more of each media type.

Media Pipeline

A mechanism for streaming and capturing audio, video, messages, and events during an Amazon Chime SDK meeting. For more information, see Creating Amazon Chime SDK media pipelines in this guide.

N

Number portability

The ability to move phone numbers between phone carriers or unified communication systems.

O

Origination

The process of receiving a call from a PSTN and handing that call off to a VoIP endpoint.

P

Participant tag

An identifier assigned to each call participant, `LEG-A` or `LEG-B`.

Policy

Amazon Chime SDK requires the following types of policies:

- **IAM user policy** – A policy that defines the permissions for Identity and Access Management users.
- **Meeting policy** – A policy that enables one user to control another user's computer when sharing screens during a meeting, and enables the option for meeting attendees to join meetings by receiving a phone call from Amazon Chime SDK.

PSTN

Public Switched Telephone Network. The infrastructure and services that provide phone calling capabilities.

PSTN Audio service

An Amazon Chime SDK service that enables developers to add audio capabilities to their communication solutions.

R

Routing

Apps created using the Amazon Chime SDK use one or more types of routing:

- **Network routing** – The process of of selecting a path for traffic in a network, or between or across multiple networks.
- **Interactions routing** – The process of ensuring that a call goes to the correct recipient or endpoint.

- **Call routing** – A call management feature that queues and distributes inbound calls to predefined recipients or endpoints.

S

SBC

Session border controller. A network element deployed to protect SIP based voice over Internet Protocol (VoIP) networks.

Sequence

The sequence of events that invoke an AWS Lambda function. Each time a function is invoked during a call, the sequence is incremented.

Service limit/service quota

The maximum number of resources, such as meetings, audio streams, or content shares, allowed by the Amazon Chime SDK For more information, see Audio in this guide.

SIP

Session Initiation Protocol, a signaling protocol used to initiate, maintain, and terminate real-time sessions that include any combination of voice, video, and messaging applications. For more information, see SIP: Session Initiation Protocol.

SIP headers

Parameters in AWS Lambda functions that contain call control data, plus other data such as user account IDs.

SIP media application

A managed object that passes values from a SIP rule to a target AWS Lambda function. Developers can call the CreateSipMediaApplication API to create SIP media applications, but they must have administrative permissions to do so.

SIP rule

A managed object that passes phone numbers for Amazon Chime SDK Voice Connector URIs to a target SIP media application.

SIP trunk

See Amazon Chime SDK Voice Connector.

SMA

See SIP media application.

SMA ID

See SIP media application.

T

Telco

A telecommunications service provider.

Termination

The process of ending a call.

Transaction

A call that contains one or more call legs. For more information, see Learn about using PSTN Audio service call legs in this guide.

Transaction ID

The ID of a transaction that contains multiple call legs. For more information, see Learn about using PSTN Audio service call legs in this guide.

V

Amazon Chime SDK Voice Connector

An object that provides provides Session Initiation Protocol (SIP) trunking service for phone systems. Administrators use the Amazon Chime SDK administrative console to create an manage Voice Connectors. For more information, see Managing Amazon Chime SDK Voice Connectors in the *Amazon Chime SDK Administrator Guide*.

Amazon Chime SDK Voice Connector group

A wrapper that contains multiple Voice Connectors from different AWS Regions. Groups allow incoming calls to fail over across Regions, which creates a fault-tolerant mechanism. For more information, see Managing Amazon Chime SDK Voice Connector groups in the *Amazon Chime SDK Administrator Guide*.

Generating insights from calls using call analytics

The topics in this section explain how to use Amazon Chime SDK call analytics to generate insights from your call data.

Amazon Chime SDK call analytics gives developers low-code solutions for generating cost-effective insights from real-time audio, including audio ingestion, analysis, alerting, and data lake integration. Call analytics enables you to generate insights through integration with Amazon Transcribe and Transcribe Call Analytics (TCA), and natively through Amazon Chime SDK voice analytics. Call analytics can also record calls to your Amazon S3 Bucket.

You can use the following methods to configure and run call analytics.

- Use the Amazon Chime SDK console to create a call analytics configuration and associate it with an Amazon Chime SDK Voice Connector. During that process, you can enable call recording and analytics. You don't need to write code to complete the process.
- Use a set of Amazon Chime SDK APIs Amazon Chime SDK APIs to programmatically create and run a configuration.

For more information, refer to Creating call analytics configurations and Using call analytics configurations, later in this section.

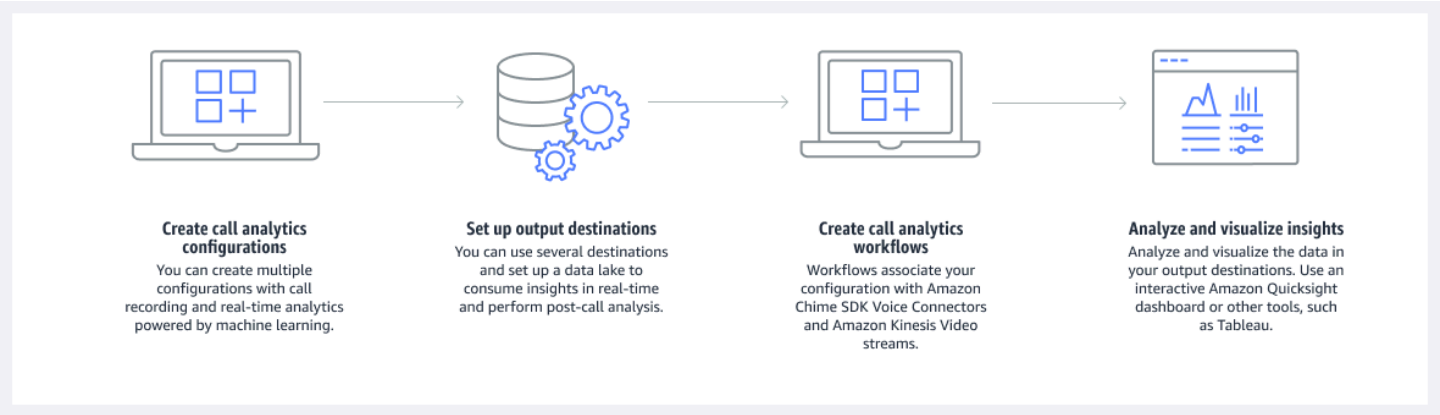
Topics

- Using Amazon Chime SDK voice analytics
- Call analytics service quotas

What is Amazon Chime SDK call analytics

Amazon Chime SDK call analytics is a low-code solution for generating cost-effective insights from real-time audio, including capabilities for audio ingestion, recording, voice analytics, alerting, and a data lake. You can generate machine learning powered insights using call analytics by creating a reusable call analytics configuration that determines which AWS machine learning integrations and audio processing features to enable for a workflow. You then use the call analytics configuration with various media sources such as Voice Connectors or Amazon Kinesis Video Streams. Call analytics generates insights through integrations with Amazon Transcribe and Transcribe call analytics (TCA), and natively through Amazon Chime SDK voice analytics, a service that runs under call analytics.

You follow these steps to use call analytics:



In the diagram:

1. You start by creating a call analytics configuration.
2. You set your output destinations and an optional data lake.
3. You create workflows that associate your configuration with a Voice Connector and Amazon Kinesis Video Streams.
4. You analyze and optionally visualize your insights.

You can use the Amazon Chime SDK console to create a call analytics configuration and enable call analytics to start automatically. If you need to control the configurations that apply to a given

type of call, you use APIs to create a configuration. Either way, the configuration contains details about the AWS machine learning services to invoke for the call audio, enable call recording, and the destinations for the insights, metadata, and recordings. Call analytics provides the following destinations:

- An Amazon Kinesis Data Stream (KDS). You can use KDS to receive live call insights that you can then integrate into your application. For instance, you can integrate the live insights to help a sales or customer-support agent during a customer call, or use the insights to augment generative AI prompts and summaries.
- An Amazon S3 bucket configured as a data warehouse. The bucket stores data in Parquet format. Parquet is an open-source file format designed to compress and store large volumes of data. You can then use Amazon Athena to query that data using simple query language (SQL), or move the data to your existing data warehouse to pair up with your business data. For example, you can perform post-call aggregate analytics to understand effectiveness of the customer calls, problems areas of a product, or opportunities to train employees to achieve better customer outcomes.

In addition to those destinations, call analytics also supports real-time alerts that you can preconfigure, based on the insights. The alerts are sent to Amazon EventBridge.

> **Note**
>
> When you create a call analytics configuration, you don't select a specific audio source. That allows you to reuse configurations across multiple audio sources. For example, a configuration can enable call recording and provide call transcription. You can then use the configuration with a Chime SDK Voice Connector and an audio stream via a Kinesis Video Stream. You can also share the configuration among multiple Voice Connectors. Each call analytics configuration is unique and identified by an ARN.

Understanding call analytics terminology

The following terminology and concepts are central to understanding how to use Amazon Chime SDK call analytics.

Amazon Athena

An interactive query service that enables you to analyze data in Amazon S3 using standard SQL. Athena is serverless, so you have no infrastructure to manage, and you pay only for the queries that you run. To use Athena, point to your data in Amazon S3, define the schema, and use standard SQL queries. You can also use workgroups to group users and control the resources they have access to when running queries. Workgroups enable you to manage query concurrency and prioritize query execution across different groups of users and workloads. For more information, refer to What is Amazon Athena.

Amazon Kinesis Data Firehose

An extract, transform, and load (ETL) service that reliably captures, transforms, and delivers streaming data to data lakes, data stores, and analytics services. For more information, refer to What Is Amazon Kinesis Data Firehose.

Call analytics data warehouse

Optional storage for call analytics data. The warehouse stores data in a parquet-based data file format in an Amazon S3 bucket. You can use standard SQL to query the data. You enable the warehouse in a call analytics configuration.

Glue Data Catalog

A centralized metadata repository for data assets across various data sources. The catalog consists of databases and tables. For call analytics, the metadata in the table tells Athena the location of your Amazon S3 bucket. It also specifies the data structure, such as column names, data types, and the table's name. Databases only hold the metadata and schema information for a dataset. For more information, refer to Understanding the AWS Glue data catalog table structure, later in this section.

Media insights pipeline

A temporary resource identified by a unique `MediaPipelineId`. Created by using a call analytics pipeline configuration and runtime parameters. The runtime parameters specify the data source for the pipeline.

Media insights pipeline configuration

A static configuration used to create media insights pipelines. You can use a configuration to instantiate one or more pipelines.

Media insights pipeline configuration element

The media insights pipeline configuration element includes instructions for either processing media using a processor element or delivering generated insights using a sink element.

Media insights pipeline task

A temporary sub-resource of a media insights pipeline. Tasks hold metadata about the status of a process for a specific stream ARN and channel ID. Identified by a unique ID. Created by starting voice analytics on a media insights pipeline.

Speaker search

A voice analytics feature that helps you to recognize call participants.

Voice analytics

An Amazon Chime SDK feature that includes speaker search and voice tone analysis.

Voice embedding

A vector representation of a caller's voice, plus a unique ID.

Voice enhancement

A system that enhances the audio quality of phone calls.

Voice profile

The combination of a voice embedding, its ID, and its expiration date.

Voice profile domain

A collection of voice profiles.

Voice tone analysis

A voice analytics feature that enables you to analyze caller voices for a `positive`, `negative`, or `neutral` sentiment.

For more information about the APIs used to create call insights configurations, initiate pipelines, and run voice analytics, refer to Amazon Chime SDK Media Pipelines, in the *Amazon Chime SDK API Reference*.

> **Note**
>
> We strongly recommend using the media insights pipeline APIs to run call analytics, because only those APIs provide new features. For more information about the differences between the media pipeline and voice namespaces, refer to Using voice APIs to run voice analytics, later in this section.

Creating call analytics configurations

To use call analytics, you start by creating a *configuration*, a static structure that holds the information needed to create a call analytics pipeline. You can use the Amazon Chime SDK console to create a configuration, or call the CreateMediaInsightsPipelineConfiguration API.

A call analytics configuration includes details about audio processors, such as recording, voice analytics, or Amazon Transcribe. It also includes insight destinations and alert event configurations. Optionally, you can save your call data to an Amazon S3 bucket for further analysis.

However, *configurations do not include specific audio sources*. That allows you reuse the configuration across multiple call analytics workflows. For example, you can use the same call analytics configuration with different Voice Connectors or across different Amazon Kinesis Video Stream (KVS) sources.

You use the configurations to create pipelines when SIP calls occur through a Voice Connector, or when new media is sent to an Amazon Kinesis Video Stream (KVS). The pipelines, in turn, process the media according to the specifications in the configuration.

You can stop a pipeline programmatically at any time. Pipelines also stop processing media when a Voice Connector call ends. In addition, you can pause a pipeline. Doing so disables calls to the underlying Amazon machine learning services and resumes them when desired. However, call recording runs while you pause a pipeline.

The following sections explain the prerequisites for creating a call analytics configuration, and how to create one.

Topics

- Using APIs to create call analytics configurations.
- Associating a configuration with a Voice Connector

Understanding the Amazon Chime SDK call analytics prerequisites

Before you create a call analytics configuration, you must have the following items. You can use the AWS console to create them:

- An Amazon Chime SDK Voice Connector. If not, refer to Creating Amazon Chime SDK Voice Connectors. You must also:
  - Enable streaming for the Voice Connector. For more information, refer to Automating the Amazon Chime SDK with EventBridge, in the *Amazon Chime SDK Administrator Guide*
  - Configure the Voice Connector to use call analytics. For more information, refer to Configuring Voice Connectors to use call analytics, in the *Amazon Chime SDK Administrator Guide*.
- Amazon EventBridge targets. If not, see Monitoring the Amazon Chime SDK with EventBridge, *Amazon Chime SDK Administrator Guide*.
- A service-linked role that allows the Voice Connector to access actions on the EventBridge targets. For more information, refer to Using the Amazon Chime SDK Voice Connector service linked role policy, in the *Amazon Chime SDK Administrator Guide*.
- An Amazon Kinesis Data Stream. If not, refer to Creating and Managing Streams, in the *Amazon Kinesis Streams Developer Guide*. Voice analytics and transcription require a Kinesis Data Stream.
- To analyze calls offline, you must create an Amazon Chime SDK data lake. To do that, refer to Creating an Amazon Chime SDK data lake, later in this guide.

Using the Amazon Chime SDK console to create call analytics configurations

After you create the prerequisites listed in the previous section, you can use the Amazon Chime SDK console to create one or more call analytics configurations. You can also use the console to associate one or more Voice Connectors with your configurations. When you complete that process, call analytics begins running with the features that you enable when you create the configuration.

You follow these steps to create a call analytics configuration:

1. Specify the configuration details, including a name and optional tags.

2. Configure your recording settings. Create a call analytics configuration that includes recording and machine-learning powered insights.
3. Configure your analytics services.
4. Select output destinations for consuming real-time insights. Create an optional data lake to perform post-call analytics.
5. Create a new service role or use an existing role.
6. Set up real-time alerts that send notifications via Amazon EventBridge when certain conditions are met.
7. Review your settings and create the configuration

After you create the configuration, you enable call analytics by associating a Voice Connector with the configuration. Once you do that, call analytics starts automatically when a call comes in to that Voice Connector. For more information, refer to Associating a configuration with a Voice Connector, later in this section.

The following sections explain how to complete each step of the process. Expand them in the order listed.

Specify configuration details

To specify configuration details

1. Open the Amazon Chime console at https://console.aws.amazon.com/chime-sdk/home.
2. In the navigation pane, under **Call Analytics**, choose **Configurations**, then choose **Create configuration**.
3. Under **Basic information**, do the following:

    a. Enter a name for the configuration. The name should reflect your use case and any tags.

    b. (Optional) Under **Tags**, choose **Add new tag**, then enter your tag keys and optional values. You define the keys and values. Tags can help you query the configuration.

    c. Choose **Next**.

Configuring recording

To configure recording

- On the **Configure recording** page, do the following:

a. Choose the **Activate call recording** checkbox. This enables recording for Voice Connector calls or KVS streams and sending the data to your Amazon S3 bucket.

b. Under **File format**, choose **WAV with PCM** for the best audio quality.

   —or—

   Choose **OGG with OPUS** to compress the audio and optimize storage.

c. (Optional) As needed, choose the **Create an Amazon S3 bucket** link and follow those steps to create an Amazon S3 bucket.

d. Enter the URI of your Amazon S3 bucket, or choose **Browse** to locate a bucket.

e. (Optional) Choose **Activate voice enhancement** to help improve the audio quality of your recordings.

f. Choose **Next**.

Understanding voice enhancement

When you create a call analytics configuration, you can enable call recording and store the recorded calls in an Amazon S3 bucket. As part of that, you can also enable voice enhancement and improve the audio quality of your stored calls. Voice enhancement only applies to recordings generated after the feature is enabled. When the voice enhancement capability is active, an enhanced recording is created in addition to the original recording, and is stored in the same Amazon S3 bucket and format. Voice enhancement will generate enhanced recordings for calls that are up to 30 minutes long. Enhanced recordings will not be generated for calls that are longer than 30 minutes.

Phone calls are narrowband-filtered and sampled at 8 KHz. Voice enhancement boosts the sampling rate from 8kHz to 16kHz and uses a machine learning model to expand the frequency content from narrowband to wideband to make the speech more natural-sounding. Voice enhancement also uses a noise reduction model called Amazon Voice Focus to help reduce background noise in the enhanced audio.

Voice enhancement also uses a noise reduction model called Voice Focus. The model helps reduce background noise in the enhanced audio. Voice enhancement applies the model to the upgraded 16 KHz audio.

> **Note**
>
> The voice enhancement feature is only supported in US East (N. Virginia) Region and US West (Oregon) Region.

Voice enhancement recordings metadata are published through your configured KDS into the existing AWS Glue data catalog table *call_analytics_recording_metadata*. To identify the original call recording record from the voice enhanced call recording, a new field called *detail-subtype* with value *VoiceEnhancement* is added to KDS notification and the glue table *call_analytics_recording_metadata*. For more information on the data warehouse schema, see Call analytics data model.

Voice enhancement file format

Note the following about enhanced recording files.

- Enhanced recordings are written to the same Amazon S3 bucket as regular recordings. You configure the destination by calling the S3RecordingSinkConfiguration or S3RecordingSinkRuntimeConfigurationAPIs, or by using the Amazon Chime SDK console.
- Enhanced recordings have **_enhanced** appended to the base file name. name.
- Enhanced recordings keep the same file format as the original recording. You configure the file format by calling the S3RecordingSinkConfiguration or S3RecordingSinkRuntimeConfiguration APIs, or by using the Amazon Chime SDK console.

The following example shows a typical file name format.

```
s3://original_file_name_enhanced.wav
```

or

```
s3://original_file_name_enhanced.ogg
```

Configure analytics services

Amazon Transcribe provides text transcriptions of calls. You can then use the transcripts to augment other machine learning services such as Amazon Comprehend or your own machine learning models.

> **Note**
>
> Amazon Transcribe also provides automatic language recognition. However, You can't use that feature with custom language models or content redaction. Also, if you use language identification with other features, you can only use the languages that those features support. For more information, refer to Language identification with streaming transcriptions, in the *Amazon Transcribe Developer Guide*.

Amazon Transcribe Call Analytics is a machine-learning powered API that provides call transcripts, sentiment, and real-time conversation insights. The service eliminates the need for note-taking, and it can enable immediate action on detected issues. The service also provides post-call analytics, such as caller sentiment, call drivers, non-talk time, interruptions, talk speed, and conversation characteristics.

> **Note**
>
> By default, post-call analytics streams call recordings to your Amazon S3 bucket. To avoid creating duplicate recordings, do not enable call recording and post-call analytics at the same time.

Finally, Transcribe Call Analytics can automatically tag conversations based on specific phrases and help redact sensitive information from audio and text. For more information on the call analytics media processors, insights generated by these processors, and output destinations see Call analytics processor and output destinations, later in this section.

To configure analytics services

1. On the **Configure analytics services** page, select the check boxes next to **Voice analytics** or **Transcription services**. You can select both items.

   Select the **Voice analytics**, checkbox to enable any combination of **Speaker search** and **Voice tone analysis**.

   Select the **Transcription services** checkbox to enable Amazon Transcribe or Transcribe Call Analytics.

   a. **To enable Speaker search**

- Select the **Yes, I agree to the Consent Acknowledgement for Amazon Chime SDK voice analytics** checkbox, then choose **Accept**.

b. To enable Voice tone analysis

- Select the **Voice tone analysis** checkbox.

c. To enable Amazon Transcribe

i. Choose the **Amazon Transcribe** button.

ii. Under **Language settings**, do either of the following:

A. If your callers speak a single language, choose **Specific language**, then open the **Language** list and select the language.

B. If your callers speak multiple languages, you can automatically identify them. Choose **Automatic language detection**.

C. Open the **Language options for automatic language identification** list and select at least two languages.

D. (Optional) Open the **Preferred language** list and specify a preferred language. When the languages you selected in the previous step have matching confidence scores, the service transcribes the preferred language.

E. (Optional) Expand **Content removal settings**, select one or more options, then choose one or more of the additional options that appear. Helper text explains each option.

F. (Optional) Expand **Additional settings**, select one or more options, then choose one or more of the additional options that appear. Helper text explains each option.

d. To enable Amazon Transcribe Call Analytics

i. Choose the **Amazon Transcribe Call Analytics** button.

ii. Open the **Language** list and select a language.

iii. (Optional) Expand **Content removal settings**, select one or more options, then choose one or more of the additional options that appear. Helper text explains each option.

iv. (Optional) Expand **Additional settings**, select one or more options, then choose one or more of the additional options that appear. Helper text explains each option.

v. (Optional) Expand **Post-call analytics settings** and do the following:

   A. Choose the **Post-call analysis** checkbox.
   B. Enter the URI of your Amazon S3 bucket.
   C. Select a content redaction type.

2. When you finish making your selections, choose **Next**.

Configure output details

After you finish the media processing steps, you select a destination for the analytics output. Call analytics provides live insights via Amazon Kinesis Data Streams, and optionally through a data warehouse in an Amazon S3 bucket of your choice. To create the data warehouse, you use a CloudFormation Template. The template helps you create the infrastructure that delivers the call metadata and insights to your Amazon S3 bucket. For more information on the data warehouse creation, refer to Creating an Amazon Chime SDK data lake, later in this section. For more information on the data warehouse schema, refer to Call analytics data model, also later in this section.

If you enabled voice analytics in the previous section, you can also add voice analytics notification destinations such as AWS Lambda, Amazon Simple Queue Service, or Amazon Simple Notification Service. The following steps explain how.

To configure output details

1. Open the **Kinesis data stream** list and select your data stream.

   > **Note**
   > If you want to visualize your data, you must select the Kinesis data stream used by the Amazon S3 bucket and Amazon Kinesis Data Firehose.

2. (Optional) Expand **Additional voice analytics notification destinations** and select any combination of AWS Lambda, Amazon SNS, and Amazon SQS destinations.
3. (Optional) Under **Analyze and visualize insights**, select the **Perform historical analysis with data lake** checkbox. For more information about data lakes, refer to Creating an Amazon Chime SDK data lake, later in this section.
4. When finished, choose **Next**.

Configure access permissions

To enable call analytics, the machine learning service and other resources must have permissions to access data media and deliver insights. You can use an existing service role or use the console to create a new role. For more information about roles, refer to Using the call analytics resource access role, later in this section.

To configure access permissions

1. On the **Configure access permissions** page, do one of the following:

    1. Select **Create and use a new service role**.
    2. In the **Service role name suffix** box, enter a descriptive suffix for the role.

    —or—

    1. Select **Use an existing service role**.
    2. Open the **Service role** list and select a role.
2. Choose **Next**.

(Optional) Configure real-time alerts

> **Important**
>
> To use real-time alerts, you must first enable Amazon Transcribe or Amazon Transcribe Analytics.

You can create a set of rules that send real-time alerts to Amazon EventBridge. When an insight generated by Amazon Transcribe or Amazon Transcribe Call Analytics matches your specified rule during an analytics session, an alert is sent. Alerts have the detail type `Media Insights Rules Matched`. EventBridge supports integration with downstream services such as Amazon Lambda, Amazon SQS, and Amazon SNS to trigger notifications for the end user or initiate other custom business logic. For more information, refer to Using Amazon EventBridge notifications, later in this section.

To configure alerts

1. Under **Real-time alerts**, choose **Active real-time alerts**.
2. Under **Rules**, select **Create rule**.
3. In the **Rule name** box, enter a name for the rule.
4. Open the **Rule type** list and select the type of rule you want to use.
5. Use the controls that appear to add keywords to the rule and apply logic, such as **mentioned** or **not mentioned**.
6. Choose **Next**.

Review and create

To create the configuration

1. Review the settings in each section. As needed choose **Edit** to change a setting.
2. Choose **Create configuration**.

Your configuration appears on the **Configurations** page of the Amazon Chime SDK console.

Using APIs to create call analytics configurations.

You can programmatically create Voice Connectors and call analytics configurations, and then associate them in order to start a call analytics workflow. This guide assumes that you know how to write the code.

The APIs that you use vary, depending on the type of workflow. For example, to record audio, you first call the CreateMediaInsightsPipelineConfiguration API to create a call analytics configuration. You then call the CreateVoiceConnector to create a Voice Connector. Finally, you associate the configuration with a Voice Connector by using the PutVoiceConnectorStreamingConfiguration API.

In contrast, to record audio with a Kinesis video stream producer, you call CreateMediaInsightsPipelineConfiguration, and then call the CreateMediaInsightsPipeline API.

For more information about using call analytics configurations to enable different workflows, refer to the workflows in Using call analytics configurations, later in this section.

Associating a configuration with a Voice Connector

After you use the console to create a call analytics configuration, you use the configuration by associating a Voice Connector with it. The Voice Connector then automatically invokes call the analytics services specified in your configuration. The Voice Connector invokes call analytics for each call.

To associate a Voice Connector

1. Open the Amazon Chime console at https://console.aws.amazon.com/chime-sdk/home.
2. In the navigation pane, under **SIP Trunking**, choose **Voice Connectors**.
3. Choose the name of the Voice Connector that you want to associate with a configuration, then choose the **Streaming** tab.
4. If it isn't already selected, choose **Start** to begin streaming to Kinesis Video Streams.
5. Under **Call Analytics**, select **Activate**, and on the menu that appears, choose your call analytics configuration ARN.
6. Choose **Save**.

> **Note**
>
> After enabling, disabling, or modifying a configuration associated with a Voice Connector, allow 5 minutes for the new settings to propagate through the service and take effect.

For more information about call analytics configurations, refer to Managing call analytics in the *Amazon Chime SDK Administrator Guide*.

For more information about using call analytics configurations to enable different workflows, refer to Using call analytics configurations, later in this section.

Using call analytics configurations

To process audio using a call analytics configuration, you must create a call analytics pipeline, also known as a media insights pipeline. The pipeline is created during a call to handle the audio and is terminated at the end of the call. Call analytics pipelines require the ARN of a call analytics configuration, and information about the audio source. The call analytics configuration includes

details about audio processors, insight destinations, and alert event configurations, *but not the audio source*. This allows you to reuse the configuration across different call analytics workflows, such as with different Voice Connectors or KVS sources. The call analytics pipeline invokes the machine learning services specified in the configuration and records the audio. You can manually or automatically stop the pipeline when the call ends.

You can use call analytics pipelines in a wide variety use cases. The following workflows show potential ways to use a call analytics configuration and pipeline.

Topics

Understanding workflows for recording calls

The topics in this section list and describe the workflows for recording calls and Kinesis Video Streams.

Recording Voice Connector calls

Use this workflow when:

- You already use, or plan to use, a Voice Connector to bring SIP media into call analytics.

> **Note**
>
> Voice Connectors support SIP and SIPREC. For more information, refer to Managing Amazon Chime SDK Voice Connectors, in the *Amazon Chime SDK Administrator Guide*.

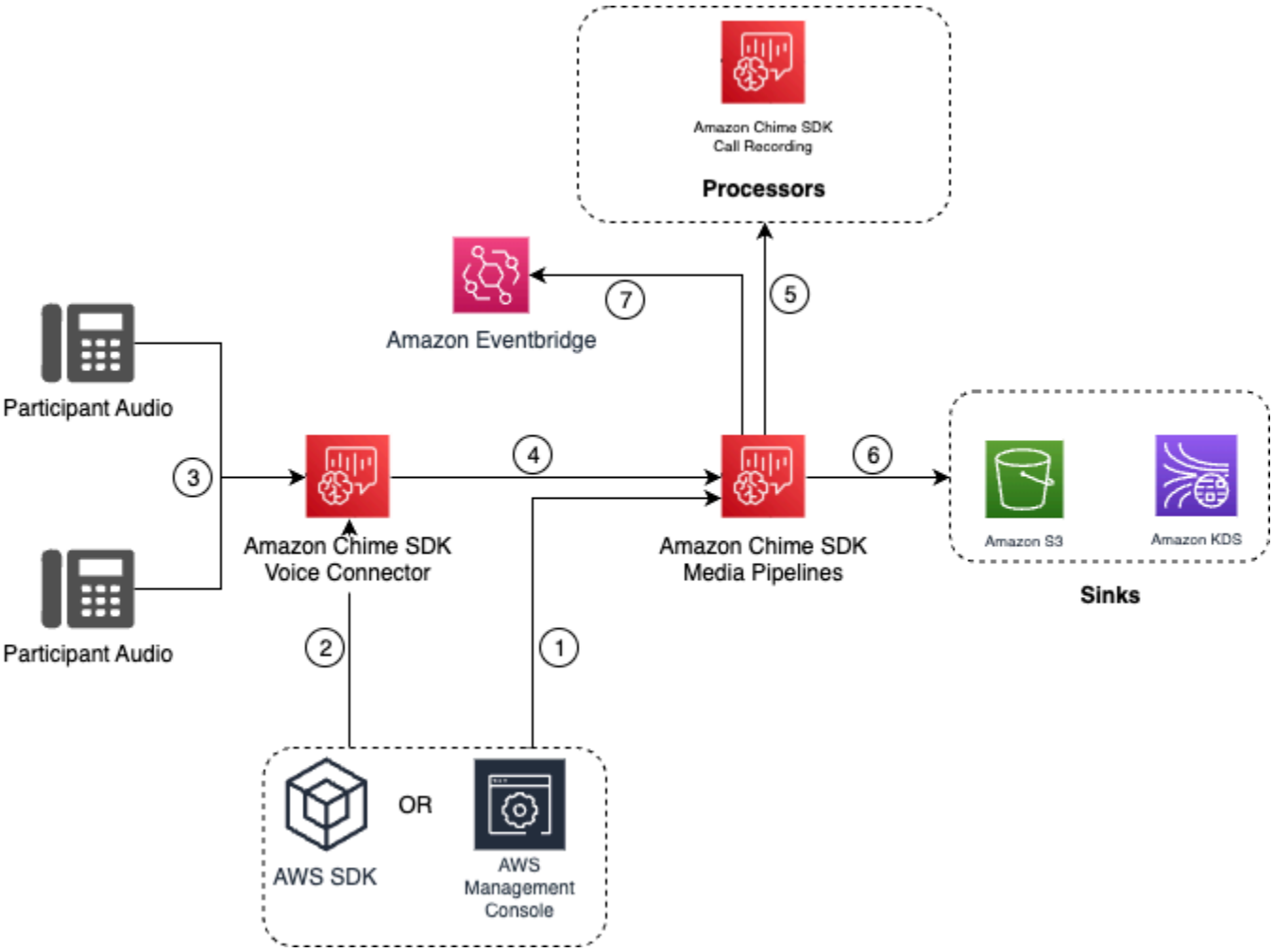
- You want to automatically record SIP or SIPREC calls with low latency to your choice of Amazon Simple Storage Service destinations.
- You want to use the Amazon Chime SDK console to create the configuration and associate it with a Voice Connector.
- You want to apply the same recording configuration to every Voice Connector call. If you want to apply multiple configurations to one or more Voice Connectors, refer to the next section.

To enable calling programmatically, use the following Amazon Chime SDK APIs.

- Use the CreateMediaInsightsPipelineConfiguration API to create a call analytics configuration
- Use the CreateVoiceConnector to create a Voice Connector.
- Use the PutVoiceConnectorStreamingConfiguration API to associate the configuration with a Voice Connector.

For more information, see Configuring Voice Connectors to use call analytics in the *Amazon Chime SDK Administrator Guide*.

The following diagram shows the flow of data when a Voice Connector initiates a call recording session. Numbers in the diagram correspond to the numbered text below.



In the diagram:

1. Use the Amazon Chime SDK console or the CreateMediaInsightsPipelineConfiguration API to create a call analytics configuration. During the configuration creation process, you simply activate call recording, choose the desired recording file format, and specify the Amazon S3

destination for storing the recording files. For more information, refer to Creating call analytics configurations, in the *Amazon Chime SDK Administrator Guide*.

2. You use the Amazon Chime SDK console or the PutVoiceConnectorStreamingConfiguration API to associate the configuration with a Voice Connector. To use the console, refer to Configuring Voice Connectors to use call analytics.
3. During an outgoing call, the Voice Connector receives each call participant's audio.
4. If a call analytics recording configuration is attached to the Voice Connector, the Voice Connector service uses the media pipeline service to initiate a call analytics recording session.
5. The media pipeline service initiates the call recording processor that monitors the ongoing call.
6. When the call ends, the media pipeline service delivers the call recording file to the designated Amazon S3 bucket and provides the recording metadata through the Amazon Kinesis Data Stream. If a data warehouse is enabled, the call metadata also goes to the Amazon Simple Storage Service data warehouse. In cases where SIPREC is utilized to incorporate SIP audio into call analytics, the call metadata includes SIPREC metadata in a table format. For more information on the recording tables, refer to Understanding the AWS Glue data catalog tables, later in this section.
7. The media pipeline service sends the pipeline status events to the default Amazon EventBridge. For more information see, Using EventBridge notifications in this guide.

> **Note**
>
> Please note, you must enable Voice Connector streaming to enable recording with a Voice Connector. This feature enables streaming of call data to the Voice Connector managed Kinesis Video Streams in your account. For more information, refer to Streaming Amazon Chime SDK Voice Connector media to Kinesis Video Streams in the *Amazon Chime SDK Administrator Guide*.

You can also to store Voice Connector created call data in Kinesis Video Streams for varying durations, ranging from hours to days or even years. Opting for no data retention limits the usability of the call data for immediate consumption. The cost of Kinesis Video Streams is determined based on the bandwidth and total storage utilized. You can adjust the data retention period at any time within the Voice Connector streaming configuration. To enable call analytics recording, you must ensure that the Kinesis Video Stream retains the data long enough to conduct the call analytics. You do that by specifying a suitable data retention period.

You can associate a call insights pipeline configuration with as many Voice Connectors as you want. You can also create a different configuration for each Voice Connector. Voice Connectors use the AWSServiceRoleForAmazonChimeVoiceConnector to call the CreateMediaInsightsPipeline API on your behalf once per transaction ID. For information about the role, see Using the Amazon Chime SDK service-linked role for Amazon Chime SDK Voice Connectors in the *Amazon Chime SDK Administrator Guide*.

Recording with Amazon Kinesis Video stream producers

You record Amazon Kinesis Video streams when:

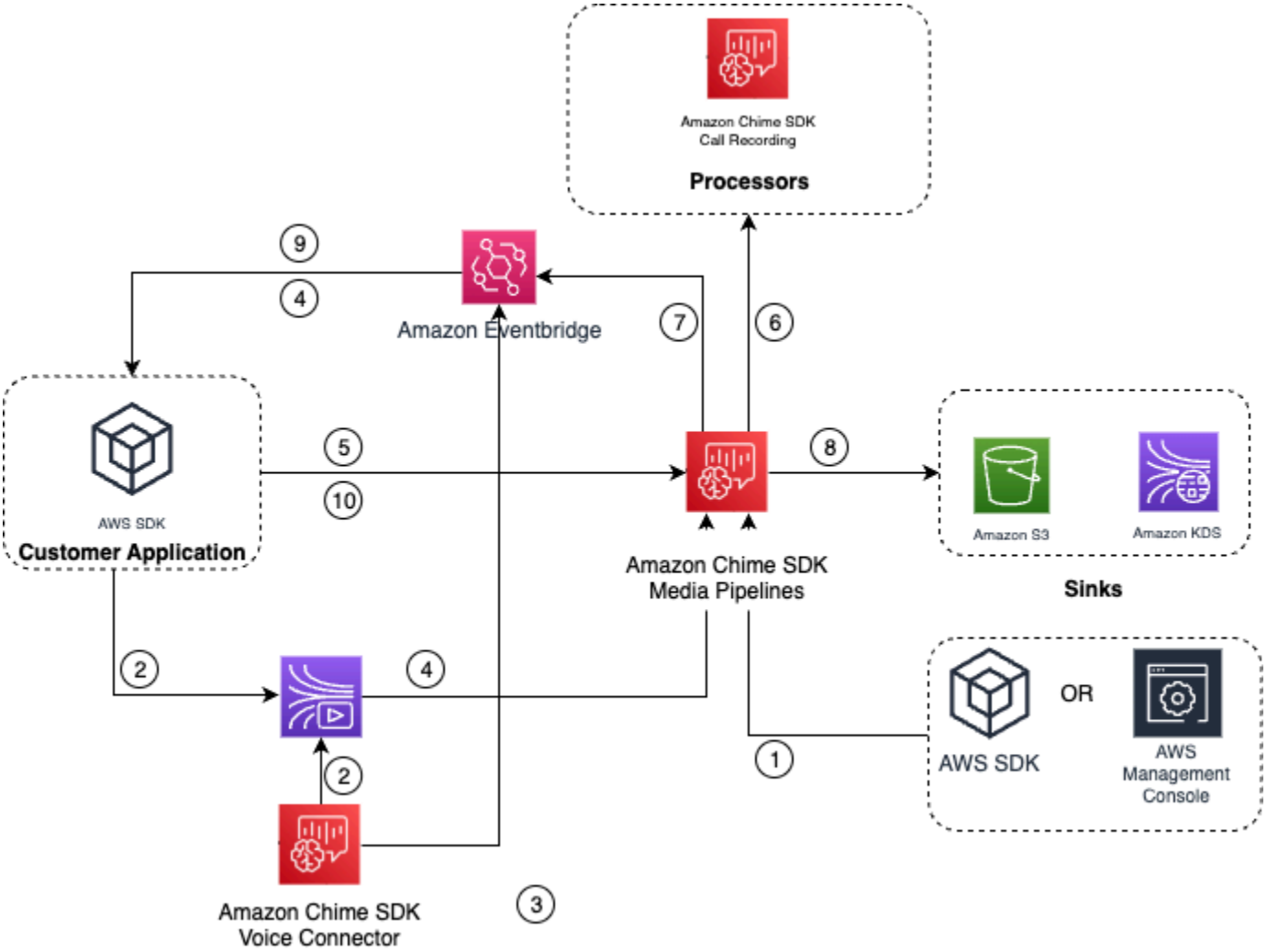
- You need to apply different configurations to a call instead of using the same configuration for every Voice Connector call.
- You want to record SIP or non-SIP audio that isn't processed by a Voice Connector.

To use this call recording option, you need to publish audio to Kinesis Video Streams (KVS) and then call the CreateMediaInsightsPipeline API with KVS stream channel information and a call analytics configuration ARN.

> **Note**
>
> The call analytics APIs support a maximum of two audio channels. You can also enable Voice Connector streaming, then use the KVS information published in the Voice Connector's EventBridge notifications to initiate a call recording.

When calling the CreateMediaInsightsPipeline API, you can choose whether or not to specify fragment numbers for each KVS stream channel definition. If you supply a fragment number, call analytics will begin processing the stream at that fragment. If you don't specify a fragment ID, call analytics begins processing the stream from the latest available fragment.

The following diagram shows the flow of data when a Voice Connector initiates a call recording session. Numbers in the diagram correspond to the numbered text below.



In the diagram:

1. You can use the Amazon Chime SDK console or the CreateMediaInsightsPipelineConfiguration API to the create a call recording configuration.
2. Use the AWS SDK to create an application that pushes external audio into KVS, or enable Voice Connector streaming to publish call audio automatically to a KVS. For more information, see Streaming Amazon Chime SDK Voice Connector media to Kinesis Video Streams in the *Amazon Chime SDK Administrator Guide*.
3. If Voice Connector streaming is enabled, the Voice Connector service sends notifications to the default EventBridge.
4. In case of Voice Connector streaming, your application can use the Amazon Chime Voice Connector streaming STARTED events from EventBridge to gather KVS stream information about the legs of a call.

5. Once your application has the audio information from Voice Connector streaming events or an external source, your application invokes the Amazon Chime SDK CreateMediaInsightsPipeline API.
6. The media pipeline service initiates the call recording processor that monitors the ongoing call.
7. The media pipeline service sends the pipeline status events to the default Amazon EventBridge. For more information, refer to Using EventBridge notifications.
8. Once a call is completed, the media pipeline service will deliver the call recording file to the designated Amazon S3 bucket and provide the recording metadata through Amazon Kinesis Data Stream. If a data warehouse is enabled, the call metadata will also be sent to the Amazon S3 data warehouse. In cases where SIPREC is utilized to incorporate SIP audio into call analytics, the call metadata will include SIPREC metadata in a convenient table format. For more information on the recording tables, refer to Understanding the AWS Glue data catalog tables, later in this section.
9. Your application can monitor the pipeline, and in case of a Voice Connector, the call status using events published to the Amazon EventBridge. For more information see, Using EventBridge notifications in this guide.
10. To terminate recording, call the DeleteMediaPipeline API to terminate the call recording.

For API based recording and examples see, Amazon S3 recording sink in this guide.

Using the CLI to start recording

The examples in this section explain how to do the following:

- Use the CLI to run a call analytics configuration and invoke the CreateMediaInsightsPipeline.
- Use the CLI to specify recording destinations audio file formats, and audio file names.

Topics

- Running a configuration and starting a pipeline
- Setting destinations, names, and formats

Running a configuration and starting a pipeline

Use the following command to run a configuration and start a media insights pipeline. The pipeline.json file contains the configuration settings.

```
aws chime-sdk-media-pipeline create-media-insights-pipeline --cli-input-json file://
pipeline.json
```

The following example shows a typical `pipeline.json` file.

```
{
    "MediaInsightsPipelineConfigurationArn": arn:aws:chime:region;account_id:media-
insights-pipeline-configuration/MyConfiguration,
    "KinesisVideoStreamRecordingSourceRuntimeConfiguration": {
        "Streams": [
            {
                "StreamArn": kinesis_video_stream_arn_1
            },
            {
                "StreamArn": kinesis_video_stream_arn_2
            }
        ],
        "FragmentSelector": {
            "FragmentSelectorType": "selector_type", // Specify "server_timestamp" or
 "producer_timestamp" as the fragment selector type
            "TimestampRange": {
                "StartTimestamp": epoch_time_seconds,
                "EndTimestamp": epoch_time_seconds
            }
        }
    },
    "S3RecordingSinkRuntimeConfiguration": {
        "Destination": arn:aws:s3:::bucket_name/prefix/optional_file_name,
        "RecordingFileFormat": file_format // Specify "Opus" or "WAV" as the recording
 file format, if you want to override the configuration
    }
}
```

The `MediaInsightsPipelineConfigurationArn` is the configuration ARN that you receive after you create a call analytics configuration.

Setting destinations, names, and formats

The following example uses a folder named `MyRecordingBucket` as the `S3SinkConfiguration.Destination` value, and `Opus` as the `RecordingFileFormat` value.

```
arn:aws:s3:::MyRecordingBucket/voice-connector-id/transaction-id_year-month-date-hour-minute-second-millisecond.ogg
```

The following example uses `MyRecordingBucket` as the `S3SinkConfiguration.Destination` value, and `Wav` as the `RecordingFileFormat` value.

```
arn:aws:s3:::MyRecordingBucket/voice-connector-id/transaction-id_year-month-date-hour-minute-second-millisecond.wav
```

Understanding workflows for machine-learning based analytics

The following sections describe how to use the machine-learning analytics features provide by Amazon Chime SDK call analytics.

> **Note**
>
> If you plan to run multiple machine-learning analytics on the same Kinesis Video Stream, you may need to increase the connection-level limit for `GetMedia` and `GetMediaForFragmentList` for the video stream. For more information, refer to Kinesis Video Streams limits in the *Kinesis Video Streams Developer Guide*.

Using Voice Connectors to initiate call analytics automatically

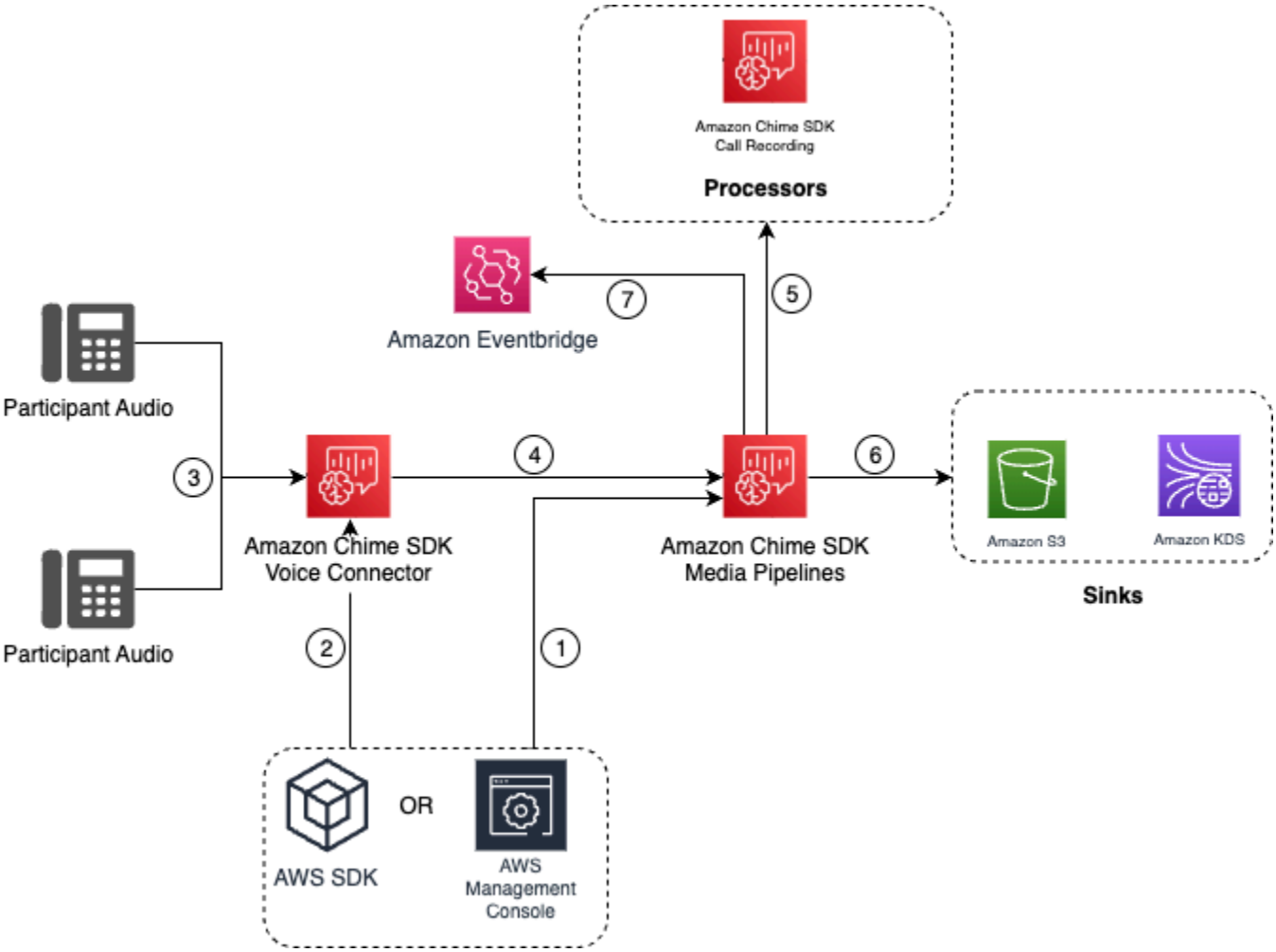
Use this workflow when:

- You want console-driven setup.
- You already use or plan to use a Voice Connector to bring SIP media into call analytics. Voice Connectors support SIP as well as SIPREC. For more information on configuring Voice Connectors, refer to Managing Amazon Chime SDK Voice Connector.
- You want to apply the same media insights configuration to every Voice Connector call.
- You need to use Amazon Chime SDK voice analytics, which requires a Voice Connector or a media insights pipeline.

To enable this workflow in the Amazon Chime SDK console, follow the steps for creating a recording configuration in Configuring Voice Connectors to use call analytics.

To enable this workflow programmatically, use the following APIs: CreateMediaInsightsPipelineConfiguration API to create a call analytics configuration and then associated the configuration to a Voice Connector using the PutVoiceConnectorStreamingConfiguration API. For more information, see Configuring Voice Connectors to use voice analytics in the *Amazon Chime SDK Administrator Guide*.

The following diagram shows the flow of data when a Voice Connector initiates a call analytics session. Numbers in the diagram correspond to the numbered text below.



In the diagram:

1. You use the Amazon Chime SDK console or the CreateMediaInsightsPipelineConfiguration API to the create a media insights pipeline configuration.
2. You use the Amazon Chime SDK console or the PutVoiceConnectorStreamingConfiguration API to associate the configuration with a Voice Connector. To associate an existing configuration

with a Voice Connector, refer to Configuring Voice Connectors to use call analytics, in the *Amazon Chime SDK Administrator Guide*.

3. During an outgoing call, the Voice Connector receives each call participant's audio.
4. Because of built-in integration with call analytics, if a call analytics configuration is attached to a Voice Connector, the Voice Connector service initiate a call analytics session using the media pipeline service.
5. The media pipeline service invokes one or more media processors as specified in the configuration.
6. The media pipeline service sends the output data to one or more destinations based on the configuration. For example, you can send real-time analytics via an Amazon Kinesis Data Stream, and if configured, you can send the call metadata and analytics to an Amazon S3 data warehouse.
7. The media pipeline service sends the pipeline status events to the default Amazon EventBridge. If you have configured rules then the notifications for them will be sent to the Amazon EventBridge as well. For more information see, Using EventBridge notifications.

Note

- A voice analytics processor only starts automatically when you call the StartSpeakerSearchTask or StartVoiceToneAnalyisTask APIs.
- You must enable Voice Connector streaming to use call analytics with Voice Connector. This feature enables streaming of call data to Voice Connector managed Kinesis Video Streams in your account. For more information, refer to Streaming Amazon Chime SDK Voice Connector media to Kinesis Video Streams in the *Amazon Chime SDK Administrator Guide*.

You can store Voice Connector call data in Kinesis Video Streams for varying amounts of time, ranging from hours to years. Opting for no data retention limits the usability of the call data for immediate consumption. The cost of Kinesis Video Streams is determined based on the bandwidth and total storage utilized. It is possible to adjust the data retention period at any time by editing your Voice Connector's streaming configuration. To enable call analytics recording, you must ensure that the Kinesis Video Stream retains data until call analytics finishes. You do that by specifying a suitable data retention period.

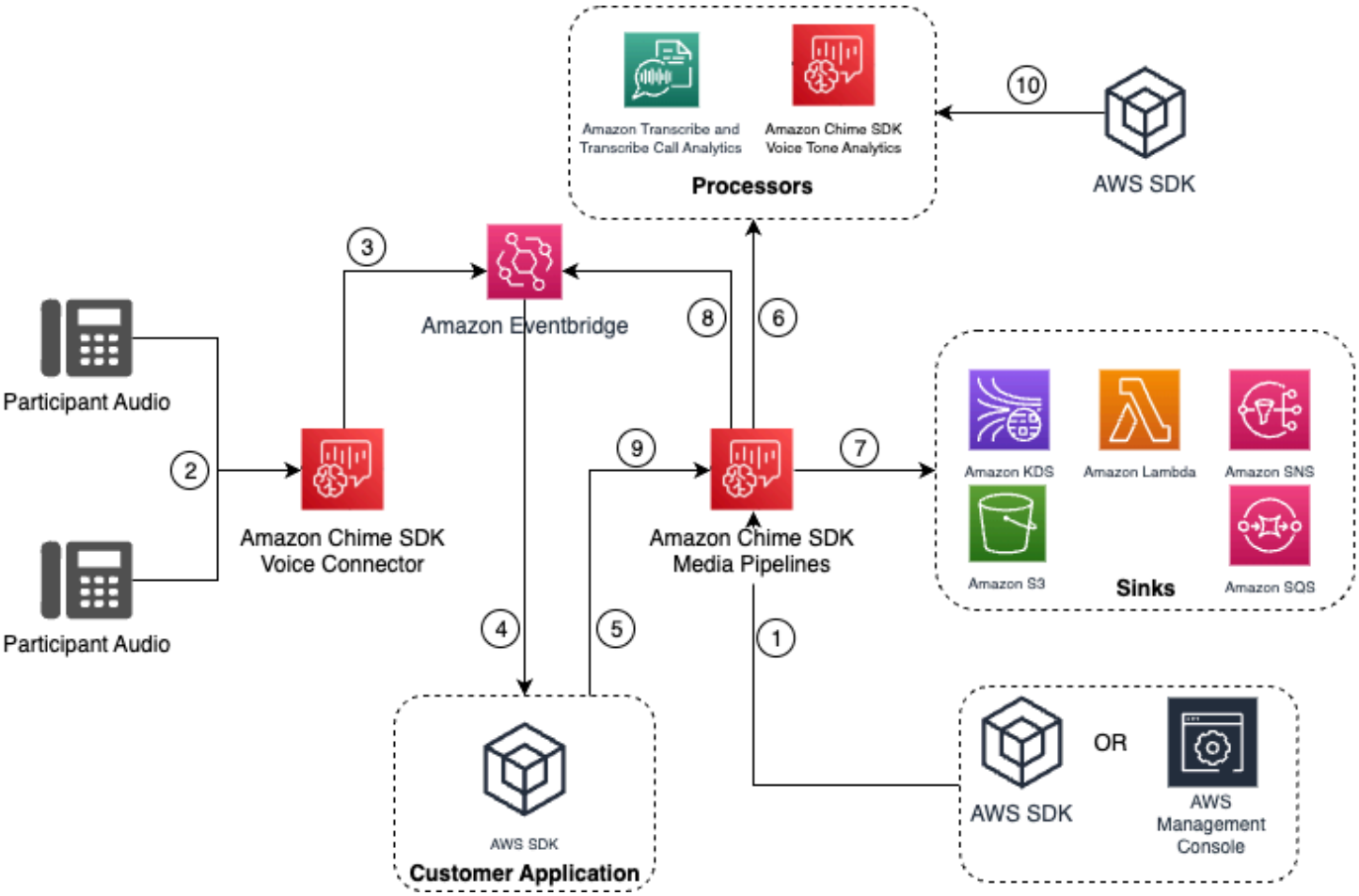
You can associate a media insights pipeline configuration with as many Voice Connectors as you want. You can also create a different configuration for each Voice Connector. Voice Connectors use the AWSServiceRoleForAmazonChimeVoiceConnector to call the CreateMediaInsightsPipeline API on your behalf once per transaction ID. For information about the role, see Using the Amazon Chime SDK service-linked role for Amazon Chime SDK Voice Connectors in the *Amazon Chime SDK Administrator Guide*.

Using call analytics APIs with Voice Connectors

Use this workflow if you use a Voice Connector but need to control when you apply a call analytics configuration and which call to apply the configuration to.

To use this method, you need to create an EventBridge target for events that the Voice Connector publishes, and then use the events to trigger the call analytics pipeline APIs. For more information, see Automating the Amazon Chime SDK with EventBridge in the *Amazon Chime SDK Administrator Guide*.

The following diagram shows how to implement more granular control when using call analytics with Voice Connector. Numbers in the diagram correspond to numbers in the text below.



In the diagram:

1. You use the Amazon Chime SDK console or the CreateMediaInsightsPipelineConfiguration API to the create a media insights pipeline configuration.
2. During an outgoing call the Voice Connector will receive participant audio.
3. The Voice Connector sends call audio to Kinesis Video Stream and corresponding events to the EventBridge. These events have stream and call metadata.
4. Your application is subscribed to EventBridge via an EventBridge Target.
5. Your application invokes the Amazon Chime SDK CreateMediaInsightsPipeline API.
6. The media pipeline service invokes one or more media processors based on the processor elements in the media insights pipeline configuration.
7. The media pipeline service sends the output data to one or more destinations based on the configuration. Amazon Chime SDK call analytics will provide real-time analytics via Amazon Kinesis Data Stream and if configured call metadata analytics to an Amazon S3 data warehouse.
8. The media pipeline service sends the events to the Amazon EventBridge. If you have configured rules then the notifications for them will be sent to the Amazon EventBridge as well.

9. You can pause or resume the call analytics session by invoking the UpdateMediaInsightsPipelineStatus API.

> **Note**
>
> Call recording does not support pausing and resuming calls. Also, voice analytics tasks started for the call also stop when you pause a session. To restart them, you must call the StartSpeakerSearchTask or StartVoiceToneAnalyisTask APIs.

10. If you select voice tone analytics during configuration, you start voice analytics by calling the StartSpeakerSearchTask or StartVoiceToneAnalyisTask APIs.

Using call analytics with Kinesis Video Streams producers

To use this option, you need to publish audio data to Kinesis Video Streams (KVS) and then call the CreateMediaInsightsPipeline API with KVS stream channel information.

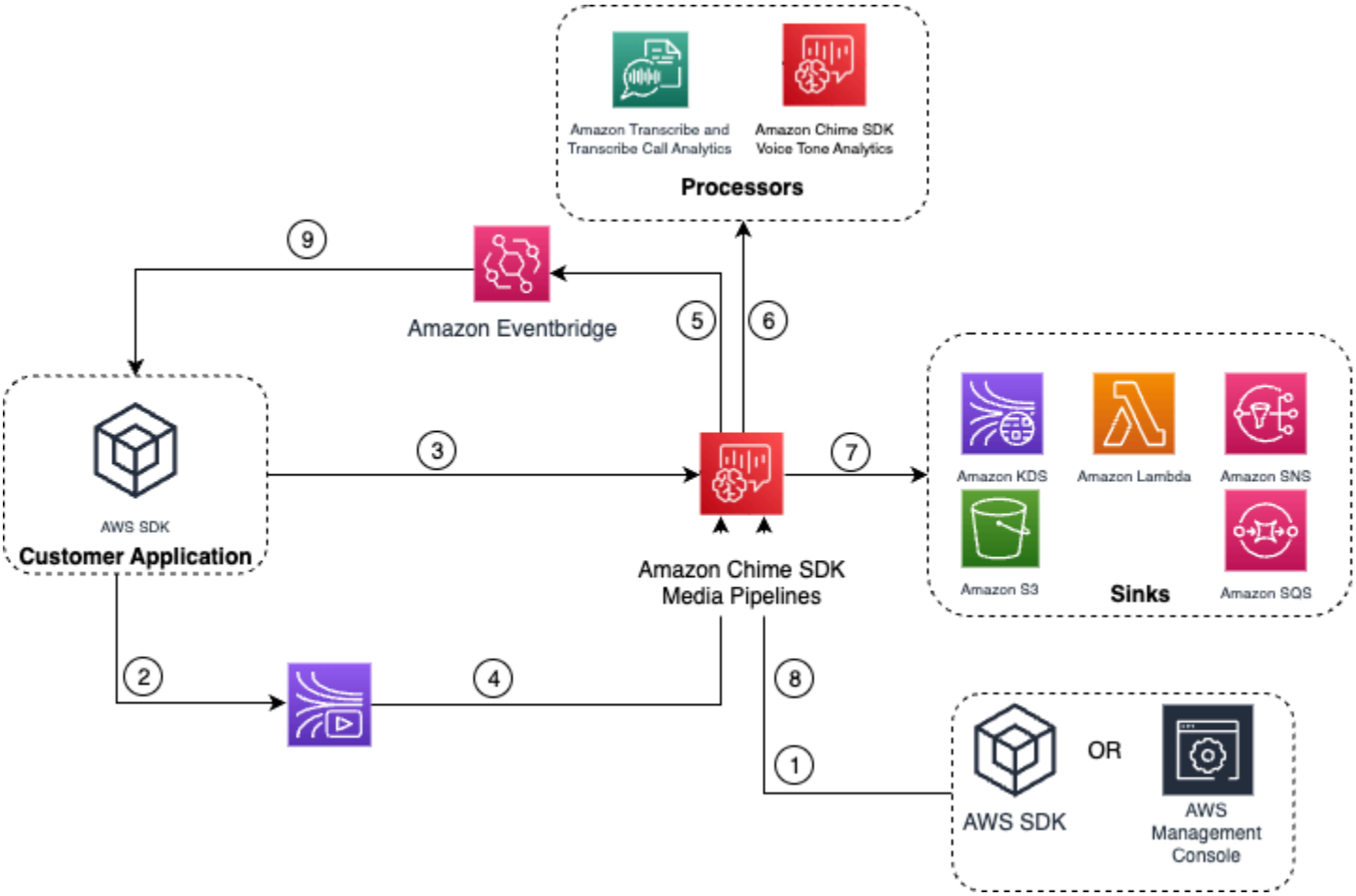
> **Note**
>
> The call analytics APIs support a maximum of two audio channels.

When calling the CreateMediaInsightsPipeline API, you can specify fragment numbers for each KVS stream channel definition. If you supply a fragment number, call analytics begins processing the stream at that fragment. Otherwise, call analytics begins processing the stream from the latest available fragment.

Call analytics supports PCM audio (only signed 16-bit little-endian audio formats, which does not include WAV) with an audio sample rate between 8kHz and 48kHz. Low-quality audio, such as telephony audio, is typically around 8,000 Hz. High-quality audio typically ranges from 16,000 Hz to 48,000 Hz. The sample rate that you specify must match that of your audio. For more information, see KinesisVideoStreamSourceRuntimeConfiguration in the *Amazon Chime SDK API Reference*.

The Kinesis Video Streams Producer SDK provides a set of libraries that you can use to stream audio data to a Kinesis Video Stream. For more information, refer to Kinesis Video Streams Producer Libraries, in the *Amazon Kinesis Video Streams Developer Guide*.

The following diagram shows the flow of data when using call analytics with a custom Kinesis Video Stream producer. Numbers in the diagram correspond to the numbered text below.



1. You use the AWS console or the CreateMediaInsightsPipelineConfiguration API to create a media insights pipeline configuration.
2. You use a Kinesis Video Stream Producer to write audio to Kinesis Video Streams.
3. Your application invokes the CreateMediaInsightsPipeline API.
4. The media pipeline service reads audio from the customer's Kinesis Video Streams.
5. The media pipeline service sends the events to the Amazon EventBridge. If you have configured rules then the notifications for them will be sent to the Amazon EventBridge as well.
6. The media pipeline service invokes one or more processor elements.
7. The media pipeline service sends output data to one or more sink elements.
8. You can pause or resume the call analytics session by invoking the UpdateMediaInsightsPipelineStatus API.

> **Note**
>
> Call recording does not support pause and resume.

9. Your application can process the Amazon EventBridge events to trigger custom business workflows.
10. If you select voice analytics when you create a configuration, your application can start voice analytics by calling the StartSpeakerSearchTask or StartVoiceToneAnalyisTask APIs.

Managing call analytics pipelines

You can read, list, and delete media insights pipelines by calling the GetMediaPipeline, ListMediaPipelines, and DeleteMediaPipeline APIs.

Media insights pipelines stop if any of the following conditions are met:

- Any of the Kinesis Video streams send no new fragments to an `InProgress` pipeline for 15 seconds.
- The DeleteMediaPipeline API is called.
- The media insights pipeline was created more than 8 hours ago. The system stops the pipeline automatically.
- The media insights pipeline is paused for more than 2 hours. The system stops the pipeline automatically.

Pausing and resuming call analytics pipelines

To pause and resume a media insights pipeline, invoke the UpdateMediaInsightsPipelineStatus API with a `Pause` or `Resume` action. To do so, you pass either the pipeline's ID or ARN in the `Identifier` field.

> **Warning**
>
> Warning: The `UpdateMediaInsightsPipelineStatus` API *stops* all voice analytics tasks started on a media insights pipeline when a `Pause` status is provided. When the `Resume` status is provided, tasks are not resumed and must be started again. You must provide all necessary notices and obtain all necessary consents from the speakers prior

to re-starting the tasks. For more information, refer to StartSpeakerSearchTask or StartVoiceToneAnalysisTask, in the *Amazon Chime SDK API Reference*.

While paused, the pipeline stops sending media to processors and writing data to Kinesis Data Streams and data warehouses. When you `Resume` the pipeline, the service sends the latest chunk available on the stream. Media insights pipelines stop automatically when paused for more than 2 hours. **Please note**, call recording does not support pause and resume.

For more, see the following topics:

- Using EventBridge notifications.
- StartSelectorType.NOW in the *Amazon Kinesis Video Streams Developer Guide*.
- Amazon Transcribe call analytics processor.

> **Note**
>
> You are billed for call analytics usage while a pipeline is paused. However, you aren't billed for AWS services accessed via the resource access role, such as Amazon Transcribe and Amazon Kinesis.

You can read, update, and delete existing call analytics configurations using GetMediaInsightsPipelineConfiguration, UpdateMediaInsightsPipelineConfiguration, and DeleteMediaInsightsPipelineConfiguration APIs by passing the configuration name or ARN in the Identifier field.

You can list configurations by calling the ListMediaInsightsPipelineConfigurations API.

Using the call analytics resource access role

The calling account must create the resource access role used by a media insights pipeline configuration. You can't use cross-account roles.

Depending on the features that you enable when you create a call analytics configuration, you must use additional resource policies. Expand the following sections to learn more.

Minimum required policy

The role requires the following policy, at a minimum:

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "transcribe:StartCallAnalyticsStreamTranscription",
                "transcribe:StartStreamTranscription"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:GetDataEndpoint",
                "kinesisvideo:GetMedia"
            ],
            "Resource": "arn:aws:kinesisvideo:us-east-1:111122223333:stream/Chime*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:GetDataEndpoint",
                "kinesisvideo:GetMedia"
            ],
            "Resource": "arn:aws:kinesisvideo:us-east-1:111122223333:stream/*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/AWSServiceName": "ChimeSDK"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": ["kms:Decrypt"],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/AWSServiceName": "ChimeSDK"
                }
            }
```

```
        }
    ]
}
```

You must also use the following trust policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "mediapipelines.chime.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
            "aws:SourceARN": "arn:aws:chime:*:111122223333:*"
        }
        }
    }
  ]
}
```

KinesisDataStreamSink policy

If you use the `KinesisDataStreamSink`, add the following policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecord"
            ],
            "Resource": [
                "arn:aws:kinesis:us-east-1:111122223333:stream/output_stream_name"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:111122223333:key/*"
            ],
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/AWSServiceName": "ChimeSDK"
                }
            }
        }
    ]
}
```

S3RecordingSink policy

If you use the `S3RecordingSink`, add the following policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectTagging",
            ],
            "Resource": [
                "arn:aws:s3:::input_bucket_path/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:GetDataEndpoint",
                "kinesisvideo:ListFragments",
                "kinesisvideo:GetMediaForFragmentList"
            ],
            "Resource": [
                "arn:aws:kinesisvideo:us-east-1:111122223333:stream/*"
            ],
```

```
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/AWSServiceName": "ChimeSDK"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:ListFragments",
                "kinesisvideo:GetMediaForFragmentList"
            ],
            "Resource": [
                "arn:aws:kinesisvideo:us-east-1:111122223333:stream/Chime*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:111122223333:key/*"
            ],
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/AWSServiceName": "ChimeSDK"
                }
            }
        }
    ]
}
```

Post Call Analytics policy

If you use the Post Call Analytics feature of the `AmazonTranscribeCallAnalyticsProcessor`, add the following policy:

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

```
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/transcribe_role_name"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "transcribe.streaming.amazonaws.com"
                }
            }
        }
    ]
}
```

VoiceEnhancementSinkConfiguration policy

If you use the `VoiceEnhancementSinkConfiguration` element, add the following policy:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectTagging"
         ],
         "Resource":[
            "arn:aws:s3:::input_bucket_path/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "kinesisvideo:GetDataEndpoint",
            "kinesisvideo:ListFragments",
            "kinesisvideo:GetMediaForFragmentList"
         ],
         "Resource":[
            "arn:aws:kinesisvideo:us-east-1:111122223333:stream/*"
```

```
            ],
            "Condition":{
               "StringLike":{
                  "aws:ResourceTag/AWSServiceName":"ChimeSDK"
               }
            }
         },
         {
            "Effect":"Allow",
            "Action":[
               "kinesisvideo:ListFragments",
               "kinesisvideo:GetMediaForFragmentList"
            ],
            "Resource":[
               "arn:aws:kinesisvideo:us-east-1:111122223333:stream/Chime*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "kms:GenerateDataKey"
            ],
            "Resource":[
               "arn:aws:kms:us-east-1:111122223333:key/*"
            ],
            "Condition":{
               "StringLike":{
                  "aws:ResourceTag/AWSServiceName":"ChimeSDK"
               }
            }
         }
      ]
   }
```

VoiceAnalyticsProcessor policy

If you use the `VoiceAnalyticsProcessor`, add the policies for `LambdaFunctionSink`, `SqsQueueSink`, and `SnsTopicSink` depending on which sinks you have defined.

`LambdaFunctionSink` policy:

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

```
  "Statement": [
    {
      "Action": [
        "lambda:InvokeFunction",
        "lambda:GetPolicy"
       ],
      "Resource": [
          "arn:aws:lambda:us-east-1:111122223333:function:function_name"
       ],
      "Effect": "Allow"
    }
  ]
}
```

SqsQueueSink policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "sqs:SendMessage",
        "sqs:GetQueueAttributes"
       ],
      "Resource": [
          "arn:aws:sqs:us-east-1:111122223333:queue_name"
        ],
      "Effect": "Allow"
    },
    {
      "Effect": "Allow",
      "Action": ["kms:GenerateDataKey", "kms:Decrypt"],
      "Resource": "arn:aws:kms:us-east-1:111122223333:key/*",
      "Condition": {
        "StringLike": {
            "aws:ResourceTag/AWSServiceName": "ChimeSDK"
        }
      }
    }
  ]
}
```

`SnsTopicSink` policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "sns:Publish",
        "sns:GetTopicAttributes"
      ],
      "Resource": [
        "arn:aws:sns:us-east-1:111122223333:topic_name"
      ],
      "Effect": "Allow"
    },
    {
      "Effect": "Allow",
      "Action": ["kms:GenerateDataKey", "kms:Decrypt"],
      "Resource": "arn:aws:kms:us-east-1:111122223333:key/*",
      "Condition": {
        "StringLike": {
            "aws:ResourceTag/AWSServiceName": "ChimeSDK"
        }
      }
    }
  ]
}
```

Understanding the call analytics statuses

Media insights pipelines track a set of statuses when you do one or both of the following:

- Use multiple machine-learning processing elements, such as Amazon Transcribe and voice analytics.
- Enable call recording with or without machine learning processing.

To get pipeline and element statuses, use the GetMediaPipeline API and EventBridge notifications.

To get statuses for voice analytics tasks, use the GetSpeakerSearchTask and GetVoiceToneAnalysisTask APIs, and voice analytics notification targets.

Media insights pipelines track the following statuses.

- **Pipeline status** – The overall status of a call analytics pipeline, also known as a media insights pipeline. This is determined by the element statuses.
- **Element status** – The processing status for the individual media insights pipeline configuration elements.
- **Task status** – The processing status for a media insights pipeline task started for voice analytics. The `VoiceAnalyticsProcessor` element status is determined by the task statuses. No other element in a call analytics pipeline has a task status.

  For more information about media insights pipeline tasks, refer to Understanding call analytics terminology, earlier in this guide.

Not all media insights configuration element types have element statuses. In general, only media insights configuration elements of the "processor" type have an element status. Also, the Amazon S3 recording and voice enhancement sinks have processor statuses. Specifically, element statuses exist for the following media insights configuration element types:

- `AmazonTranscribeProcessor`
- `AmazonTranscribeCallAnalyticsProcessor`
- `S3RecordingSink`
- `VoiceAnalyticsProcessor`
- `VoiceEnhancementSink`

The pipeline status is determined by the element statuses as follows:

Pipeline status	Condition
NotStarted	All element statuses are not started.
Initializing	At least one element is initializing and the rest are not started.
InProgress	At least one element is in progress.

Pipeline status	Condition
Failed	At least one element has failed and the remaining elements are stopped.
Stopping	Refer to Managing call analytics pipelines for a complete list of stopping conditions.
Stopped	All elements are stopped.
Paused	All elements are paused.

Unlike other element statuses, the `VoiceAnalyticsProcessor` element has a few nuances. As mentioned earlier, the `VoiceAnalyticsProcessor` element status, corresponding to the Amazon Chime SDK voice analytics feature, is determined by the task statuses created from StartSpeakerSearchTask and StartVoiceToneAnalysisTask.

- The `VoiceAnalyticsProcessor`'s element status begins in a `NotStarted` state, because `StartSpeakerSearchTask` and `StartVoiceToneAnalysisTask` must be called before the element can change the status to `Initializing`, and then `InProgress`.
- The `VoiceAnalyticsProcessor` stays `InProgress` as long as one task is started and a stop condition is not met while the task is running.
- Even though the `VoiceAnalyticsProcessor` may be `InProgress`, you will only be charged for the duration that the tasks process.
- To clean-up media insights pipelines that had at least one voice analytics task started and no more tasks running, you must call `DeleteMediaPipeline`.
- As long as a task runs or completes successfully, the `VoiceAnalyticsProcessor` element status remains at `InProgress`.

Monitoring call analytics pipelines with Amazon CloudWatch

You can use Amazon CloudWatch to monitor Amazon Chime SDK call analytics pipelines. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information about CloudWatch, see the Amazon CloudWatch User Guide.

Topics

Prerequisites

To use CloudWatch metrics, you must first create a media pipelines service-linked role that grants permissions to publish service metrics to Amazon CloudWatch. For more information about the service-linked role, see Creating a service-linked role for media pipelines, in this guide.

Call analytics metrics

Amazon Chime SDK call analytics publishes the following metrics to the `AWS/ChimeSDK` namespace for media insights pipelines that you create by using a media insights configuration.

Metric	Description
`MediaInsightsPipelineCreated`	The media insights pipeline was successfully created. Unit: Count
`MediaInsightsPipelineStopped`	The media insights pipeline was successfully stopped. Unit: Count
`MediaInsightsPipelineFailed`	The media insights pipeline failed. Unit: Count
`MediaInsightsPipelineDuration`	The time between pipeline creation and Stopped/Failed. Unit: Seconds
`MediaInsightsPipelineBillingDuration`	The billing duration of the media insights pipeline.

Metric	Description
	Unit: Count
`RecordingFileSize`	The size of the recording file. Unit: Bytes
`RecordingDuration`	The duration of the recording. Unit: Seconds

CloudWatch dimensions for pipeline metrics

The following table lists the CloudWatch dimensions that you can use to monitor call analytics pipelines.

Dimension	Description
`MediaInsightsPipelineConfigurationId`	The ID of the media insights pipeline configuration.
`MediaInsightsPipelineConfigurationName`	The name of the media insights pipeline configuration.

Call analytics processor and output destinations

You can only specify unique elements once per media insights pipeline configuration. All processors and sinks must reside in the same AWS account, and you must create them in the same AWS Region as the endpoint that you call. For example, if you use the `us-east-1` endpoint for Amazon Chime SDK Media Pipelines, you can't pass a Kinesis Data Stream from the `us-west-2` region.

Expand each section for information about each destination.

Amazon Transcribe Call Analytics processor destinations

Supported sinks: `KinesisDataStreamSink`.

You can't combine this processor with an Amazon Transcribe processor. For more information about Amazon Transcribe Call Analytics, refer to Real-time call analytics, in the *Amazon Transcribe Developer Guide*. If you enable Post call analytics by including `PostCallAnalyticsSettings` in the `AmazonTranscribeCallAnalyticsProcessorConfiguration` API call, you receive artifacts in the specified Amazon S3 location when the media insights pipeline stops and processing finishes.

> **Note**
>
> If you pause the pipeline for more than 35 seconds and then resume it, post-call artifacts are generated in separate files with different session IDs in the Amazon S3 bucket.

Post-call artifacts include an analytics JSON file and audio recording WAV or Opus file. The Amazon S3 bucket URL for redacted (if you enable content redaction) and non-redacted recording files is sent to the Kinesis Data Stream once for each Amazon Transcribe call analytics Post Call session as part of `onetimeMetadata` in the metadata section.

Call analytics with Amazon Transcribe call analytics takes audio data input from the Kinesis Video Stream.

- Supported media encoding: PCM signed 16-bit little-endian audio.
- Supported media sample rates: Between 8,000 Hz and 48,000 Hz.

`StreamConfiguration` input for an Amazon Transcribe Analytics process:

- You must specify the `KinesisVideoStreamArn` for each stream.
- (Optional) The KVS `FragmentNumber` starts a call analytics job with the chunk after a specified fragment. If not provided, it uses the latest chunk on the Kinesis video stream.
- The `StreamChannelDefinition` defines who's talking. Amazon Transcribe call analytics requires two-channel audio. You must specify which speaker is on which channel when you call the CreateMediaInsightsPipeline API. For example, if your agent speaks first, you set the `ChannelId` to `0` to indicate the first channel, and `ParticipantRole` to `AGENT` to indicate that the agent is speaking.

> **Note**
>
> When you use a Voice Connector to create a `MediaInsightsPipeline` with an Amazon Transcribe call analytics processor, the Voice Connector account leg audio is `AGENT` and the PSTN leg audio is `CUSTOMER` for the `ParticipantRole`.
> For Voice Connector SIPREC, we rely on the SIPREC metadata. In most cases, the stream label with the lowest lexicographic value is considered to be the `AGENT`.

The following example shows Kinesis Video Stream input for one dual-channel audio stream.

```
"StreamChannelDefinition" : {
    "NumberOfChannels" : 2
    "ChannelDefinitions": [
        {
            "ChannelId": 0,
            "ParticipantRole": "AGENT"
        },
        {
            "ChannelId": 1,
            "ParticipantRole": "CUSTOMER"
        }
    ]
}
```

In contrast, the following example shows two mono inputs from two different Kinesis Video streams.

```
KVS-1:
    "StreamChannelDefinition" : {
        "NumberOfChannels"  : 1
        "ChannelDefinitions": [
            {
                "ChannelId": 0,
               "ParticipantRole": "AGENT"
            }
        ]
    }
KVS-2:
    "StreamChannelDefinition" : {
        "NumberOfChannels"  : 1
```

```
        "ChannelDefinitions": [
            {
                "ChannelId": 1,
               "ParticipantRole": "CUSTOMER"
            }
        ]
    }
```

Amazon Transcribe call analytics output

Each Amazon Transcribe record contains an `UtteranceEvent` or a `CategoryEvent`, but not both. `CategoryEvents` have a `detail-type` of `TranscribeCallAnalyticsCategoryEvent`.

The following example shows one-time metadata output format for Amazon Transcribe.

```
{
    "time": "string", // ISO8601 format
    "service-type": "CallAnalytics",
    "detail-type": "CallAnalyticsMetadata",
    "mediaInsightsPipelineId": "string",
    "metadata": "string" // JSON encoded string of the metadata object
}

// metadata object
{
    "voiceConnectorId": "string",
    "callId": "string",
    "transactionId": "string",
    "fromNumber": "string",
    "toNumber": "string",
    "direction": "string",
    "oneTimeMetadata": "string" // JSON encoded string of oneTimeMetadata object
}

// onetimeMetadata object
{
    "inviteHeaders": "string", // JSON encoded string of SIP Invite headers key-value
 pair
    "siprecMetadata": "string", // siprec metadata in XML
    "siprecMetadataJson": "string", // siprec metadata in JSON (converted from above
 XML)

    // If PostcallSettings are enabled for Amazon Transcribe Call Analytics
```

```
    "s3RecordingUrl": "string",
    "s3RecordingUrlRedacted": "string"
}

// inviteHeaders object
{
    "string": "string"
}
```

The following example shows the Amazon Transcribe Call Analytics output format.

```
{
    "time": "string", // ISO8601 format
    "service-type": "CallAnalytics",
    "detail-type": "TranscribeCallAnalytics",
    "mediaInsightsPipelineId": "string",
    "metadata": {
        "voiceConnectorId": "string",
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "direction": "string"
    },
    "UtteranceEvent": {
        "UtteranceId": "string",
        "ParticipantRole": "string",
        "IsPartial": boolean,
        "BeginOffsetMillis": number,
        "EndOffsetMillis": number,
        "Transcript": "string",
        "Sentiment": "string",
        "Items": [{
            "Content": "string",
            "Confidence": number,
            "VocabularyFilterMatch": boolean,
            "Stable": boolean,
            "ItemType": "string",
            "BeginOffsetMillis": number,
            "EndOffsetMillis": number,
        }, ]
        "Entities": [{
            "Content": "string",
```

```
            "Confidence": number,
            "Category": "string", // Only PII is supported currently
            "Type": "string",
            "BeginOffset": number,
            "EndOffset": number,
        }, ],
        "IssuesDetected": [{
            "CharacterOffsets": {
                "Begin": number,
                "End": number
            }
        }]
    },
    "CategoryEvent": {
        "MatchedCategories": ["string"],
        "MatchedDetails": {
            "string": {
                "TimestampRanges": [{
                    "BeginOffsetMillis": number,
                    "EndOffsetMillis": number
                }]
            }
        }
    }
}
```

Amazon Chime SDK Voice Connector streaming updates metadata

If the call analytics configuration is associated with an Amazon Chime SDK Voice Connector, the following Voice Connector Update payload will be sent when there is a Voice Connector streaming update.

The following example shows an update metadata format for Amazon Transcribe processor and Transcribe Call Analytics processor.

```
{
    "time": "string", // ISO8601 format
    "service-type": "CallAnalytics",
    "detail-type": "CallAnalyticsMetadata",
    "callevent-type": "Update",
    "metadata": "string" // JSON encoded string of the metadata object
}
```

```
// metadata object
{
    "voiceConnectorId": "string",
    "callId": "string",
    "transactionId": "string",
    "fromNumber": "string",
    "toNumber": "string",
    "direction": "string",
    "oneTimeMetadata": "string" // JSON encoded string of oneTimeMetadata object
}

// onetimeMetadata object
{
    "sipHeaders": "string", // JSON encoded string of SIP Invite headers key-value pair
    "siprecMetadata": "string", // siprec metadata in XML
    "siprecMetadataJson": "string" // siprec metadata in JSON (converted from above
 XML)
}

// sipHeaders object
{
    "string": "string"
}
```

The following example shows an update metadata format for Call Analytics Amazon S3 Recording.

```
{
    "time": "string", // ISO8601 format
    "service-type": "CallAnalytics",
    "detail-type": "Recording",
    "callevent-type": "Update",
    "metadata": "string" // JSON encoded string of the metadata object
}

// metadata object
{
    "voiceConnectorId": "string",
    "callId": "string",
    "transactionId": "string",
    "fromNumber": "string",
    "toNumber": "string",
    "direction": "string",
    "oneTimeMetadata": "string" // JSON encoded in string of oneTimeMetadata object
```

```
}

// onetimeMetadata object
{
    "sipHeaders": "string", // JSON encoded string of SIP Invite headers key-value pair
    "siprecMetadata": "string", // siprec metadata in XML
    "siprecMetadataJson": "string" // siprec metadata in JSON (converted from above
 XML)
}

// sipHeaders object
{
    "string": "string"
}
```

SIP call recording metadata

The following examples show the metadata for recording a SIP call between two people, Alice and Bob. Both participants send and receive audio and video. For simplicity, the example only has snippets of SIP and SDP, and SRC records the streams of each participant to SRS without mixing.

```
INVITE sip:recorder@example.com SIP/2.0
   Via: SIP/2.0/TCP src.example.com;branch=z9hG4bKdf6b622b648d9
   From: <sip:2000@example.com>;tag=35e195d2-947d-4585-946f-09839247
   To: <sip:recorder@example.com>
   Call-ID: d253c800-b0d1ea39-4a7dd-3f0e20a
   Session-ID: ab30317f1a784dc48ff824d0d3715d86
    ;remote=00000000000000000000000000000000
   CSeq: 101 INVITE
   Max-Forwards: 70
   Require: siprec
   Accept: application/sdp, application/rs-metadata,
   application/rs-metadata-request
   Contact: <sip:2000@src.example.com>;+sip.src
   Content-Type: multipart/mixed;boundary=boundary
   Content-Length: [length]

   Content-Type: application/SDP
   ...
   m=audio 49170 RTP/AVP 0
   a=rtpmap:0 PCMU/8000
   a=label:96
   a=sendonly
```

```
   ...
   m=video 49174 RTP/AVPF 96
   a=rtpmap:96 H.264/90000
   a=label:97
   a=sendonly
   ...
   m=audio 51372 RTP/AVP 0
   a=rtpmap:0 PCMU/8000
   a=label:98
   a=sendonly
   ...
   m=video 49176 RTP/AVPF 96
   a=rtpmap:96 H.264/90000
   a=label:99
   a=sendonly
   ....

Content-Type: application/rs-metadata
Content-Disposition: recording-session

<?xml version="1.0" encoding="UTF-8"?>
<recording xmlns='urn:ietf:params:xml:ns:recording:1'>
  <datamode>complete</datamode>
        <group group_id="7+OTCyoxTmqmqyA/1weDAg==">
                <associate-time>2010-12-16T23:41:07Z</associate-time>
                <!-- Standardized extension -->
                <call-center xmlns='urn:ietf:params:xml:ns:callcenter'>
                        <supervisor>sip:alice@atlanta.com</supervisor>
                </call-center>
                <mydata xmlns='http://example.com/my'>
                        <structure>structure!</structure>
                        <whatever>structure</whatever>
                </mydata>
        </group>
        <session session_id="hVpd7YQgRW2nD22h7q60JQ==">
                <sipSessionID>ab30317f1a784dc48ff824d0d3715d86;
                                        remote=47755a9de7794ba387653f2099600ef2</
sipSessionID>
                <group-ref>7+OTCyoxTmqmqyA/1weDAg==
                </group-ref>
                <!-- Standardized extension -->
                <mydata xmlns='http://example.com/my'>
                        <structure>FOO!</structure>
                        <whatever>bar</whatever>
```

```
                </mydata>
        </session>
        <participant
              participant_id="srfBElmCRp2QB23b7Mpk0w==">
                <nameID aor="sip:alice@atlanta.com">
                        <naSRCme xml:lang="it">Alice</name>
                </nameID>
                <!-- Standardized extension -->
                <mydata xmlns='http://example.com/my'>
                        <structure>FOO!</structure>
                        <whatever>bar</whatever>
                </mydata>
        </participant>
        <participant
               participant_id="zSfPoSvdSDCmU3A3TRDxAw==">
                <nameID aor="sip:bob@biloxy.com">
                        <name xml:lang="it">Bob</name>
                </nameID>
                <!-- Standardized extension -->
                <mydata xmlns='http://example.com/my'>
                        <structure>FOO!</structure>
                        <whatever>bar</whatever>
                </mydata>
        </participant>
        <stream stream_id="UAAMm5GRQKSCMVvLyl4rFw=="
               session_id="hVpd7YQgRW2nD22h7q60JQ==">
                <label>96</label>
        </stream>
        <stream stream_id="i1Pz3to5hGk8fuXl+PbwCw=="
               session_id="hVpd7YQgRW2nD22h7q60JQ==">
                <label>97</label>
        </stream>
        <stream stream_id="8zc6e0lYTlWIINA6GR+3ag=="
               session_id="hVpd7YQgRW2nD22h7q60JQ==">
                <label>98</label>
        </stream>
        <stream stream_id="EiXGlc+4TruqqoDaNE76ag=="
               session_id="hVpd7YQgRW2nD22h7q60JQ==">
                <label>99</label>
        </stream>
        <sessionrecordingassoc session_id="hVpd7YQgRW2nD22h7q60JQ==">
            <associate-time>2010-12-16T23:41:07Z</associate-time>
        </sessionrecordingassoc>
        <participantsessionassoc
```

```
              participant_id="srfBElmCRp2QB23b7Mpk0w=="
              session_id="hVpd7YQgRW2nD22h7q60JQ==">
                <associate-time>2010-12-16T23:41:07Z</associate-time>
        </participantsessionassoc>
        <participantsessionassoc
               participant_id="zSfPoSvdSDCmU3A3TRDxAw=="
               session_id="hVpd7YQgRW2nD22h7q60JQ==">
                <associate-time>2010-12-16T23:41:07Z</associate-time>
        </participantsessionassoc>
        <participantstreamassoc
              participant_id="srfBElmCRp2QB23b7Mpk0w==">
                <send>i1Pz3to5hGk8fuXl+PbwCw==</send>
                <send>UAAMm5GRQKSCMVvLyl4rFw==</send>
                <recv>8zc6e0lYTlWIINA6GR+3ag==</recv>
                <recv>EiXGlc+4TruqqoDaNE76ag==</recv>
        </participantstreamassoc>
        <participantstreamassoc
               participant_id="zSfPoSvdSDCmU3A3TRDxAw==">
                <send>8zc6e0lYTlWIINA6GR+3ag==</send>
                <send>EiXGlc+4TruqqoDaNE76ag==</send>
                <recv>UAAMm5GRQKSCMVvLyl4rFw==</recv>
                <recv>i1Pz3to5hGk8fuXl+PbwCw==</recv>
        </participantstreamassoc>
</recording>
```

The following example shows the updated metadata when one call participant puts the other on hold. In this case, `participant_id srfBElmCRp2QB23b7Mpk0w==` only receives media streams and does not send any media, so the `send` XML element is omitted. In contrast, `participant_id zSfPoSvdSDCmU3A3TRDxAw==` sends media to, but does not receive media from, the other participant, so the `recv` XML element is omitted.

```
INVITE sip:recorder@example.com SIP/2.0
            Via: SIP/2.0/TCP src.example.com;branch=z9hG4bKdf6b622b648d9
      From: <sip:2000@example.com>;tag=35e195d2-947d-4585-946f-09839247
      To: <sip:recorder@example.com>
      Call-ID: d253c800-b0d1ea39-4a7dd-3f0e20a
      Session-ID: ab30317f1a784dc48ff824d0d3715d86
       ;remote=f81d4fae7dec11d0a76500a0c91e6bf6
      CSeq: 101 INVITE
      Max-Forwards: 70
      Require: siprec
      Accept: application/sdp, application/rs-metadata,
```

```
      application/rs-metadata-request
      Contact: <sip:2000@src.example.com>;+sip.src
      Content-Type: multipart/mixed;boundary=foobar
      Content-Length: [length]

      Content-Type: application/SDP
      ...
      m=audio 49170 RTP/AVP 0
      a=rtpmap:0 PCMU/8000
      a=label:96
      a=sendonly
      ...
      m=video 49174 RTP/AVPF 96
      a=rtpmap:96 H.264/90000
      a=label:97
      a=sendonly
      ...
      m=audio 51372 RTP/AVP 0
      a=rtpmap:0 PCMU/8000
      a=label:98
      a=sendonly
      ...
      m=video 49176 RTP/AVPF 96
      a=rtpmap:96 H.264/90000
      a=label:99
      a=sendonly
      ....

   Content-Type: application/rs-metadata
   Content-Disposition: recording-session

      <?xml version="1.0" encoding="UTF-8"?>
        <recording xmlns='urn:ietf:params:xml:ns:recording:1'>
          <datamode>partial</datamode>
            <participantstreamassoc
             participant_id="srfBElmCRp2QB23b7Mpk0w==">
             <recv>8zc6e0lYTlWIINA6GR+3ag==</recv>
             <recv>EiXGlc+4TruqqoDaNE76ag==</recv>
            </participantstreamassoc>
            <participantstreamassoc
             participant_id="zSfPoSvdSDCmU3A3TRDxAw==">
              <send>8zc6e0lYTlWIINA6GR+3ag==</send>
              <send>EiXGlc+4TruqqoDaNE76ag==</send>
             </participantstreamassoc>
```

```
            </recording>
```

The following example shows the metadata update when the call resumes. The payload now has the `send` and `recv` XML elements.

```
INVITE sip:recorder@example.com SIP/2.0
      Via: SIP/2.0/TCP src.example.com;branch=z9hG4bKdf6b622b648d9
      From: <sip:2000@example.com>;tag=35e195d2-947d-4585-946f-09839247
      To: <sip:recorder@example.com>
      Call-ID: d253c800-b0d1ea39-4a7dd-3f0e20a
      Session-ID: ab30317f1a784dc48ff824d0d3715d86
       ;remote=f81d4fae7dec11d0a76500a0c91e6bf6
      CSeq: 101 INVITE
      Max-Forwards: 70
      Require: siprec
      Accept: application/sdp, application/rs-metadata,
      application/rs-metadata-request
      Contact: <sip:2000@src.example.com>;+sip.src
      Content-Type: multipart/mixed;boundary=foobar
      Content-Length: [length]

      Content-Type: application/SDP
      ...
      m=audio 49170 RTP/AVP 0
      a=rtpmap:0 PCMU/8000
      a=label:96
      a=sendonly
      ...
      m=video 49174 RTP/AVPF 96
      a=rtpmap:96 H.264/90000
      a=label:97
      a=sendonly
      ...
      m=audio 51372 RTP/AVP 0
      a=rtpmap:0 PCMU/8000
      a=label:98
      a=sendonly
      ...
      m=video 49176 RTP/AVPF 96
      a=rtpmap:96 H.264/90000
      a=label:99
      a=sendonly
      ....
```

```
Content-Type: application/rs-metadata
Content-Disposition: recording-session

   <?xml version="1.0" encoding="UTF-8"?>
     <recording xmlns='urn:ietf:params:xml:ns:recording:1'>
       <datamode>partial</datamode>
         <participantstreamassoc
          participant_id="srfBElmCRp2QB23b7Mpk0w==">
          <send>i1Pz3to5hGk8fuXl+PbwCw==</send>
          <send>UAAMm5GRQKSCMVvLyl4rFw==</send>
          <recv>8zc6e0lYTlWIINA6GR+3ag==</recv>
          <recv>EiXGlc+4TruqqoDaNE76ag==</recv>
         </participantstreamassoc>
         <participantstreamassoc
          participant_id="zSfPoSvdSDCmU3A3TRDxAw==">
           <send>8zc6e0lYTlWIINA6GR+3ag==</send>
           <send>EiXGlc+4TruqqoDaNE76ag==</send>
           <recv>i1Pz3to5hGk8fuXl+PbwCw==</recv>
          <recv>UAAMm5GRQKSCMVvLyl4rFw==</recv>
          </participantstreamassoc>
        </recording>
```

Amazon Transcribe processor destinations

Supported sinks: `KinesisDataStreamSink`.

You can't combine this processor with the Amazon Transcribe call analytics. For more information on the input to, and output of, Amazon Transcribe, refer to Transcribe streaming audio in the *Amazon Transcribe Developer Guide*.

Call analytics session with Amazon Transcribe takes audio data input from Kinesis Video Stream.

- Supported MediaEncoding: PCM signed 16-bit little-endian audio.
- Supported MediaSampleRate sample rates: Between 8,000 Hz and 48,000 Hz.

`StreamConfiguration` input for Amazon Transcribe processors:

- You must specify the `KinesisVideoStreamArn` for each stream.
- (Optional) KVS `FragmentNumber` - Starts a call analytics job with the chunk after a specific fragment. If not provided, it will use the latest available chunk on the Kinesis Video Stream.

- `StreamChannelDefinition` Amazon Transcribe currently supports audio with two channels. You have to specify the `NumberOfChannels` in the runtime `StreamChannelDefinition`. Additionally, you must pass the `ChannelId` if you send mono audio in two separate channels. In your transcript, channels are assigned the labels `ch_0` and `ch_1`. The following example shows the KVS input for one mono audio channel stream.

```
"StreamChannelDefinition" : {"
    NumberOfChannels" : 1
}
```

The following example shows the KVS input for two mono audio inputs in two different streams.

```
KVS-1:
    "StreamChannelDefinition" : {
        "NumberOfChannels"  : 1
        "ChannelDefinitions": [
            {
                "ChannelId": 0
            }
        ]
    }
KVS-2:
    "StreamChannelDefinition" : {
        "NumberOfChannels"  : 1
        "ChannelDefinitions": [
            {
                "ChannelId": 1
            }
        ]
    }
```

Note

For the Voice Connector created `MediaInsightsPipeline` with an Amazon Transcribe processor, the Voice Connector account leg audio is assigned to `channel-0` and the PSTN leg audio to `channel-1`.
For Voice Connector SIPREC, we rely on the SIPREC metadata. In most cases, the stream label with the lowest lexicographic value is assigned to `channel-0`.

For the Amazon Transcribe and Amazon Transcribe call analytics processors, if you pass two Kinesis Video streams, and each stream contains a mono audio channel, we interleave both channels to a single audio stream before processing Transcribe or Transcribe call analytics data.

Amazon Transcribe output

The following example shows a one-time metadata output format for Amazon Transcribe.

```
{
    "time": "string", // ISO8601 format
    "service-type": "CallAnalytics",
    "detail-type": "CallAnalyticsMetadata",
    "mediaInsightsPipelineId": "string",
    "metadata": "string" // JSON encoded string of the metadata object
}

// metadata object
{
    "voiceConnectorId": "string",
    "callId": "string",
    "transactionId": "string",
    "fromNumber": "string",
    "toNumber": "string",
    "direction": "string",
    "oneTimeMetadata": "string" // JSON encoded string of oneTimeMetadata object
}

// onetimeMetadata object
{
    "inviteHeaders": "string", // JSON encoded string of SIP Invite headers key-value
 pair
    "siprecMetadata": "string", // siprec metadata in XML
    "siprecMetadataJson": "string" // siprec metadata in JSON (converted from above
 XML)
}

// inviteHeaders object
{
    "string": "string"
}
```

The following example shows the Amazon Transcribe output format.

```
{
    "time": "string", // ISO8601 format
    "service-type": "CallAnalytics",
    "detail-type": "Transcribe",
    "mediaInsightsPipelineId": "string",
    "metadata": {
        "voiceconnectorId": "string",
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "direction": "string"
    }
    "TranscriptEvent": {
        "Transcript": {
            "Results": [{
                "Alternatives": [{
                    "Entities": [{
                        "Category": "string",
                        "Confidence": number,
                        "Content": "string",
                        "EndTime": number,
                        "StartTime": number,
                        "Type": "string"
                    }],
                    "Items": [{
                        "Confidence": number,
                        "Content": "string",
                        "EndTime": number,
                        "Speaker": "string",
                        "Stable": boolean,
                        "StartTime": number,
                        "Type": "string",
                        "VocabularyFilterMatch": boolean
                    }],
                    "Transcript": "string"
                }],
                "ChannelId": "string",
                "EndTime": number,
                "IsPartial": boolean,
                "LanguageCode": "string",
                "LanguageIdentification": [{
```

```
                    "LanguageCode": "string",
                    "Score": number
                }],
                "ResultId": "string",
                "StartTime": number
            }]
        }
    }
}
```

Voice analytics processor destinations

Supported sinks: `KinesisDataStreamSink`, `SqsQueueSink`, `SnsTopicSink`, and `LambdaFunctionSink`.

You can combine this processor with the Amazon Transcribe call analytics processor, the Amazon Transcribe processor, or call recording. You must use the StartSpeakerSearchTask or StartVoiceToneAnalysisTask APIs to invoke a voice analytics processor. For more information about using voice analytics, refer to Using Amazon Chime SDK voice analytics.

Using Kinesis Data Stream as a sink

The Kinesis Data Stream (KDS) records generated by call analytics include the media pipeline ID, detail type, metadata, and processor-specific sections. For information about consuming data from a Kinesis Data Stream, refer to Reading Data from Amazon Kinesis Data Streams, in the *Amazon Kinesis Streams Developer guide*. To create a configuration with this sink, you must have `kinesis:DescribeStream` permission on the specified stream.

Metadata

The `metadata` section of the generated KDS records contains any key-value pairs specified in `CallAnalyticsRuntimeMetadata` during the CreateMediaInsightsPipeline API call. If a call analytics session was initiated by a Voice Connector, the metadata section is automatically populated with the following parameters:

- `transactionId`
- `fromNumber`
- `toNumber`
- `callId`
- `voiceConnectorId`

- `direction`

In addition to the parameters shown above, the metadata section for Voice Connector initiated call analytics sessions will be populated with a `oneTimeMetadata` field that contains:

- `inviteHeaders`
- `siprecMetadata`

This is published to Kinesis Data Streams only once at the beginning of the session and has a `detail-type` of `CallAnalyticsMetadata`.

You can pass unique identifiers in the `MediaInsightsRuntimeMetadata` for each CreateMediaInsightsPipeline API call so that you can uniquely identify the source of each record delivered to your Kinesis Data Stream.

Amazon S3 call recording

Call analytics recording reads audio from a KVS stream, records it as an audio file, and uploads the file to the specified Amazon S3 Bucket. After recording call analytics also sends the call metadata along with the file location to KDS. If you enable a data warehouse, the call metadata (including SIPREC metadata if SIPREC was used) is delivered to the data warehouse in a set of Parquet tables that you can query.

Like any other call analytics processor, you need to first create a configuration for the pipeline. You can use the Amazon Chime SDK Console or the CLI to create the configuration. You then use the CLI to create the pipeline. For more information on using the console to create recording configurations, refer to Creating call analytics configurations, earlier in this section. For more information about using the recording workflows, refer to Understanding workflows for recording calls, earlier in this section.

To use the CLI to create a configuration

Run the following command:

```
aws chime-sdk-media-pipeline create-media-insights-pipeline-configuration --cli-input-
json file://configuration.json
```

The following example shows a configuration JSON file with only recording enabled:

```
{
    "MediaInsightsPipelineConfigurationName": configuration_name,
    "ResourceAccessRoleArn": role_arn,
    "Elements": [
        {
            "KinesisDataStreamSinkConfiguration": {
                "InsightsTarget": KDS_arn //Where recording live metadata will be
 delivered.
            },
            "Type": "KinesisDataStreamSink"
        },
        {
            "S3RecordingSinkConfiguration": {
                "Destination": "arn:aws:s3:::kvs-recording-testing",
                "RecordingFileFormat": file_format // Specify "Opus" or "WAV" as the
 recording file format.
            },
            "Type": "S3RecordingSink"
        }
    ]
}
```

Remember the following:

- To enable call recording through Kinesis Video Streams, audio should be PCM signed 16-bit little-endian. The sample rate must be 8KHz.
- Builders must set a long enough data retention period for the Kinesis Video Stream to ensure the fragments are retained and consumable by call analytics.
- If you enable call recording, either by itself or in combination with other processors, you must supply two Kinesis Video Stream ARNs for recording. Call recording does not support a single stereo audio input.

Amazon S3 call recording metadata output

The following example shows metadata output format for call analytics Amazon S3 recording.

```
{
    "time": "string", // ISO8601 format
    "service-type": "CallAnalytics",
    "detail-type": "Recording",
```

```
    "mediaInsightsPipelineId": "string",
    "s3MediaObjectConsoleUrl": "string",
    "recordingDurationSeconds": "number",
    "metadata": "string" // JSON encoded string of the metadata object
}

// metadata object
{
    "voiceConnectorId": "string",
    "callId": "string",
    "transactionId": "string",
    "fromNumber": "string",
    "toNumber": "string",
    "direction": "string",
    "startTime": "string", // ISO8601 format
    "endTime": "string", // ISO8601 format
    "oneTimeMetadata": "string" // JSON encoded in string of oneTimeMetadata object
}

// onetimeMetadata object
{
    "sipHeaders": "string", // JSON encoded string of SIP Invite headers key-value pair
    "siprecMetadata": "string", // siprec metadata in XML
    "siprecMetadataJson": "string" // siprec metadata in JSON (converted from above
 XML)
}

// sipHeaders object
{
    "string": "string"
}
```

Enable voice enhancement

To enable voice enhancement, include a `VoiceEnhancementSinkConfiguration` element in a CreateMediaInsightsPipelineConfiguration API call.

This example shows a typical element.

```
{
  "Type":"VoiceEnhancementSink",
  "VoiceEnhancementSinkConfiguration": {
       "Disabled": Boolean (string) // FALSE ==> Voice Enhancement will be performed
```

```
}
```

To update a configuration, add the `VoiceEnhancementSinkConfiguration` element to a UpdateMediaInsightsPipelineConfiguration API call. When you do, the GetMediaInsightsPipelineConfiguration API includes the `VoiceEnhancementSinkConfiguration` element in the results.

This example request shows how to enable Voice Enhancement and Amazon S3 recording.

```
POST /media-insights-pipeline-configurations HTTP/1.1
Content-type: application/json

{
   "MediaInsightsPipelineConfigurationName":"media_insights_configuration_name",
   "ResourceAccessRoleArn":"arn:aws:iam::account_id:role/resource_access_role",
   "Elements":[
      {
         "Type":"S3RecordingSink",
         "S3RecordingSinkConfiguration":{
            "Destination":"arn:aws:s3:::input_bucket_path",
            "RecordingFileFormat":"Wav"
         }
      },
      {
         "Type":"VoiceEnhancementSink",
         "VoiceEnhancementSinkConfiguration": {
            "disabled":"false"
         }
      }
   ],
   "ClientRequestToken":"client_request_token"
}
```

> **Note**
>
> The `VoiceEnhancementSink` element always requires an `S3RecordingSink` element in a call analytics configuration.

Combining transcription with recording sinks

You can combine Amazon Transcribe and Amazon Transcribe Call Analytics processors with an Amazon S3 recording sink. Builders can pass an S3RecordingSinkConfiguration in addition to the Amazon Transcribe processors in a CreateMediaInsightsPipelineConfiguration API call, or by using the console.

In conjunction with the Amazon S3 recording sink, you can use an Amazon Transcribe or an Amazon Transcribe Call Analytics processor, but never both. You can also add voice analytics to the same configuration in addition to a recording sink, with or without a transcribe processor.

> **Note**
>
> You can enable recording with any of the processors listed above. However, if you enable Amazon Transcribe Call Analytics along with Amazon S3 call recording, you must provide two Kinesis video streams, and you'll receive duplicate recording files, one from Amazon Transcribe Call Analytics and one from the Amazon S3 call recording.

Remember the following:

- You must use a unique `MediaInsightsPipelineConfigurationName`.
- For information about the `ResourceAccessRoleArn`, refer to Using the call analytics resource access role in this guide.
- The `Destination` value must be an S3 path ARN. The Amazon S3 bucket must be owned by the same account.
- If you use a configuration with Transcribe and recording to create a pipeline, pauses and resumes only appear in the insights generated by a Kinesis Data stream. All the data in the KVS streams is recorded and uploaded to Amazon S3.
- If a configuration uses Amazon transcribe or transcribe call analytics (TCA) in addition to recording, the media insights pipeline provides transcription or Transcribe Call Analytics insights in real time, followed by Amazon S3 recording at the end of the call. If transcribe services fail during the call analytics, the S3 recording job still tries to run. Conversely, an Amazon S3 recording failure doesn't affect the transcribe insights, since it runs after transcribe completes.

This example shows a configuration with an Amazon Transcribe processor and an Amazon S3 recording sink. The example also enables partial results stabilization, which can reduce latency in

output, but may impact accuracy. For more information, refer to Partial-result stabilization, in the *Amazon Transcribe Developer Guide*.

```
{
    "MediaInsightsPipelineConfigurationName": unique_configuration_name,
    "ResourceAccessRoleArn": role_arn,
    "Elements": [{
            "AmazonTranscribeProcessorConfiguration": {
                "ContentIdentificationType": "string",
                "ContentRedactionType": "string",
                "EnablePartialResultsStabilization": boolean, //Enables partial result
 stabilization. Can reduce latency. May impact accuracy.
                "FilterPartialResults": boolean, //To control partial utterance events
                "LanguageCode": "string",
                "LanguageModelName": "string",
                "PartialResultsStability": "string",
                "PiiEntityTypes": "string",
                "ShowSpeakerLabel": boolean,
                "VocabularyFilterMethod": "string",
                "VocabularyFilterName": "string",
                "VocabularyName": "string"
            },
            "Type": "AmazonTranscribeProcessor"
        },
        {
            "KinesisDataStreamSinkConfiguration": {
                "InsightsTarget": KDS_arn //Where recording and insights live metadata
 will be delivered.
            },
            "Type": "KinesisDataStreamSink"
        },
        {
            "S3RecordingSinkConfiguration": {
                "Destination": S3_Arn,
                "RecordingFileFormat": file_format // Specify "Opus" or "WAV" as the
 recording file format.
            },
            "Type": "S3RecordingSink"
        }
    ]
}
```

Using Amazon EventBridge notifications

Amazon Chime SDK Call Analytics supports sending events to the default EventBridge bus when the state of the media insights pipeline changes, or when call analytics real-time alert conditions are met. For the media insights pipeline error status updates, we recommend that you configure an EventBridge target to notify you if your resources fail asynchronously. Call analytics notifications have a aws.chime source and various detail types, which are shared in the following sections. For more information, see the Amazon EventBridge User Guide.

Topics

Status updates

Media insights pipelines send EventBridge notifications as a call analytics session progresses and either ends successfully or encounters errors. You receive an EventBridge notification with the “Media Insights State Change" detail type when:

- The status of a media insights pipeline changes.
- The status of a media insights pipeline element changes.
- Any pipeline element is stopped.
- Any pipeline element fails.

The detail section always includes the following fields:

- `version`
- `mediaInsightsPipelineArn`
- `eventType`

The detail section also includes a `mediaInsightsPipelineElementStatuses` field if the media insights pipeline contains multiple elements, such as analytics processors and data sinks. This field indicates the statuses of each element in the pipeline. The possible status for each pipeline element could be:

- `NotStarted`
- `InProgress`

- `Stopped`
- `Failed`

The detail section also includes any key-value pairs specified in `MediaInsightsRuntimeMetadata` during the CreateMediaInsightsPipeline API call. If a call analytics session was initiated by a Voice Connector, the metadata section is populated with the following parameters automatically:

- `transactionId`
- `fromNumber`
- `toNumber`
- `callId`
- `voiceConnectorId`
- `direction`

The following event types may appear whenever a media insights pipeline contains a single element. Expand each section to learn more.

Amazon Chime SDK media insights in progress

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": []
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsInProgress",
        "version": "0",
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
```

```
        "voiceConnectorId": "string",
        "direction": "string"
    }
}
```

Amazon Chime SDK media insights paused

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": []
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsPaused",
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string"
    }
}
```

Amazon Chime SDK media insights stopped

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
```

```
    "resources": []
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsStopped",
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string"
    }
}
```

Amazon Chime SDK media insights temporary failure

Indicates that the service encountered a temporary failure and will attempt to retry. No action is required from you.

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": []
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsTemporaryFailure",
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string"
    }
}
```

Amazon Chime SDK media insights permanent failure

Indicates a failure that requires action from you. Use the `failureReason` to troubleshoot the problem. Typical failures could include the following:

- Insufficient permissions on the resource access role
- Missing or deleted resources
- Throttling from an AWS service that call analytics invokes on your behalf, such as Amazon Transcribe or Amazon Kinesis.
- Incompatible media formats on KVS streams

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": []
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsPermanentFailure",
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string",
        "failureReason": "string"
    }
}
```

> **Note**
>
> The `failureReason` field is optional. For example, a typical reason could be `Access denied when assuming resource access role`.

The following event types may appear whenever a media insights pipeline is created, or the creation attempt fails, for a call analytics session initiated by an Amazon Chime SDK Voice Connector. Expand each section to learn more.

Amazon Chime SDK media insights created

This example shows a typical success event.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": []
    "detail":
    {
        "version": "0",
        "mediaInsightsPipelineConfigurationArn": "string",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsCreated",
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string",
    }
}
```

Amazon Chime SDK media insights create failed

This example shows a typical failure event.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": []
    "detail":
    {
        "version": "0",
        "mediaInsightsPipelineConfigurationArn": "string",
        "eventType": "chime:MediaInsightsCreateFailed",
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string",
        "failureOrigin": "Voice Connector",
        "httpStatusCode": "string",
        "failureReason": "string"
    }
}
```

The following event types may appear when a media insights pipeline contains multiple elements. The example notifications are for `AmazonTranscribeProcessor` combined with `S3RecordingSink`. Expand each section to learn more.

AmazonTranscribeProcessor is in progress and S3RecordingSink has not started

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": [],
```

```
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsInProgress",
        "mediaInsightsPipelineElementStatuses": [
            {
                "type": "AmazonTranscribeProcessor",
                "status": "InProgress",
                "updatedOn": 1686184070655
            },
            {
                "type": "S3RecordingSink",
                "status": "NotStarted",
                "updatedOn": 1686184070655
            }
        ]
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string"
    }
}
```

AmazonTranscribeProcessor has succeeded and S3RecordingSink is in progress

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": [],
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsInProgress",
        "mediaInsightsPipelineElementStatuses": [
            {
```

```
                "type": "AmazonTranscribeProcessor",
                "status": "Stopped",
                "updatedOn": 1686184070655
            },
            {
                "type": "S3RecordingSink",
                "status": "InProgress",
                "updatedOn": 1686184070655
            }
        ]
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string"
    }
}
```

AmazonTranscribeProcessor has failed and S3RecordingSink is in progress

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": [],
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsInProgress",
        "mediaInsightsPipelineElementStatuses": [
            {
                "type": "AmazonTranscribeProcessor",
                "status": "Failed",
                "updatedOn": 1686184070655
            },
            {
                "type": "S3RecordingSink",
```

```
                "status": "InProgress",
                "updatedOn": 1686184070655
            }
        ]
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string"
    }
}
```

AmazonTranscribeProcessor has failed and S3RecordingSink has succeeded

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": [],
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsPermanentFailure",
        "mediaInsightsPipelineElementStatuses": [
            {
                "type": "AmazonTranscribeProcessor",
                "status": "Failed",
                "updatedOn": 1686184070655
            },
            {
                "type": "S3RecordingSink",
                "status": "Stopped",
                "updatedOn": 1686184070655
            }
        ]
        "callId": "string",
        "transactionId": "string",
```

```
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string",
        "failureReason": "string"
    }
}
```

AmazonTranscribeProcessor has succeeded and S3RecordingSink has failed

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": [],
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsPermanentFailure",
        "mediaInsightsPipelineElementStatuses": [
            {
                "type": "AmazonTranscribeProcessor",
                "status": "Stopped",
                "updatedOn": 1686184070655
            },
            {
                "type": "S3RecordingSink",
                "status": "Failed",
                "updatedOn": 1686184070655
            }
        ]
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string",
        "failureReason": "string"
```

```
    }
}
```

AmazonTranscribeProcessor is paused and S3RecordingSink has not started

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": [],
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsPaused",
        "mediaInsightsPipelineElementStatuses": [
            {
                "type": "AmazonTranscribeProcessor",
                "status": "Paused",
                "updatedOn": 1686184070655
            },
            {
                "type": "S3RecordingSink",
                "status": "NotStarted",
                "updatedOn": 1686184070655
            }
        ]
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string"
    }
}
```

AmazonTranscribeProcessor has temporarily failed and S3RecordingSink has not started

This example shows a typical event structure.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": [],
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsTemporaryFailure",
        "mediaInsightsPipelineElementStatuses": [
            {
                "type": "AmazonTranscribeProcessor",
                "status": "TemporarilyFailed",
                "updatedOn": 1686184070655
            },
            {
                "type": "S3RecordingSink",
                "status": "NotStarted",
                "updatedOn": 1686184070655
            }
        ]
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string"
    }
}
```

AmazonTranscribeProcessor and S3RecordingSink succeeded

This example shows a typical event structure.

```
{
```

```
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights State Change",
    "source": "aws.chime",
    "account": number,
    "region": "string",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": [],
    "detail": {
        "version": "0",
        "mediaInsightsPipelineArn": "string",
        "eventType": "chime:MediaInsightsStopped",
        "mediaInsightsPipelineElementStatuses": [
            {
                "type": "AmazonTranscribeProcessor",
                "status": "Stopped",
                "updatedOn": 1686184070655
            },
            {
                "type": "S3RecordingSink",
                "status": "Stopped",
                "updatedOn": 1686184070655
            }
        ]
        "callId": "string",
        "transactionId": "string",
        "fromNumber": "string",
        "toNumber": "string",
        "voiceConnectorId": "string",
        "direction": "string"
    }
}
```

S3RecordingSink succeeded and VoiceEnhancement in progress

This example shows a typical event structure.

```
{
  "version": "0",
  "id": "string",
  "detail-type": "Media Insights State Change",
  "source": "aws.chime",
  "account": number,
  "time": "yyyy-mm-ddThh:mm:ssZ",
```

```
  "region": "string",
  "detail": {
    "mediaInsightsPipelineArn": "string",
    "eventType": "chime:MediaInsightsInProgress",
    "version": "0",
    "mediaInsightsPipelineElementStatuses": [
      {
        "type": "VoiceEnhancementSink",
        "status": "InProgress",
        "updatedOn": 1686184070655
      },
      {
        "type": "S3RecordingSink",
        "status": "Stopped",
        "updatedOn": 1686184070655
      }
    ]
  }
}
```

S3RecordingSink succeeded and VoiceEnhancement failed due to calls longer than 30 minutes

This example shows a typical event structure.

```
{
  "version": "0",
  "id": "string",
  "detail-type": "Media Insights State Change",
  "source": "aws.chime",
  "account": number,
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "string",
  "detail": {
    "mediaInsightsPipelineArn": "string",
    "eventType": "chime:MediaInsightsStopped",
    "version": "0",
    "mediaInsightsPipelineElementStatuses": [
      {
        "type": "VoiceEnhancement",
        "status": "NotSupported",
        "updatedOn": 1686184070655,
        "statusDetail": "Unsupported recording length"
      },
      {
```

```
        "type": "S3RecordingSink",
        "status": "Stopped",
        "updatedOn": 1686184070655
      }
    ]
  }
}
```

S3RecordingSink succeeded and VoiceEnhancement failed due to calls less than 30 minutes

This example shows a typical event structure.

```
{
  "version": "0",
  "id": "string",
  "detail-type": "Media Insights State Change",
  "source": "aws.chime",
  "account": number,
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "string",
  "detail": {
    "mediaInsightsPipelineArn": "string",
    "eventType": "chime:MediaInsightsPermanentFailure",
    "version": "0",
    "mediaInsightsPipelineElementStatuses": [
      {
        "type": "VoiceEnhancement",
        "status": "Failed",
        "updatedOn": 1686184070655
      },
      {
        "type": "S3RecordingSink",
        "status": "Stopped",
        "updatedOn": 1686184070655
      }
    ]
  }
}
```

Real-time alerts

> **Note**
>
> Only Amazon Transcribe and Amazon Transcribe Call Analytics processors support real-time alerts.

Amazon Chime SDK call analytics allows developers to set up rules for sending real-time alerts through a processor during an analytics session. Alerts are sent to Amazon EventBridge with the detail type `Media Insights Rules Matched`. EventBridge supports integration with downstream services such as Lambda, Amazon SQS, and Amazon SNS to trigger notifications for the end user or initiate other custom business logic.

Real-time alerts are set up as a part of the `RealTimeAlertConfiguration` field for the `MediaInsightsPipelineConfiguration`. You can use the Amazon Chime SDK console to configure the field, or you can call the CreateMediaInsightsPipelineConfiguration or UpdateMediaInsightsPipelineConfiguration APIs.

This example shows how to create or update a real-time alert configuration through the API.

```
{
    "MediaInsightsPipelineConfigurationName": "config_name",
    "ResourceAccessRoleArn": "arn:aws:iam::account_id:role/role_name",
    "RealTimeAlertConfiguration": {
        "Disabled": false,
        "Rules": [{
                "Type": "KeywordMatch",
                "KeywordMatchConfiguration": {
                    "RuleName": "rule_name_1",
                    "Keywords": [
                        "hello",
                        "thank you"
                    ],
                    "Negate": false
                }
            },
            {
                "Type": "Sentiment",
                "RuleName": "rule_name_2",
                "SentimentType": "NEGATIVE",
```

```
                "TimePeriod": 60
            },
            {
                "Type": "IssueDetection",
                "RuleName": "rule_name_3"
            }
        ]
    },
    "Elements": [{
            "Type": "AmazonTranscribeCallAnalyticsProcessor",
            "AmazonTranscribeCallAnalyticsProcessorConfiguration": {
                "LanguageCode": "en-US"
            }
        },
        {
            "Type": "KinesisDataStreamSink",
            "KinesisDataStreamSinkConfiguration": {
                "InsightsTarget": "arn:aws:kinesis:us-
east-1:account_id:stream/stream_name"
            }
        }
    ]
}
```

Each rule in a real-time alert configuration is triggered independently. You may receive multiple EventBridge notifications if multiple rule conditions are met at the same time. To create a list of rules for your alerts, you can select among the following rule types:

Keyword Match

Alerts when a specified set of keyword or phrases are matched in an utterance or transcript event. You can configure the alert to emit an event if:

- Any specified keywords are spoken, and `Negate` is set to `false`.
- All specified keywords are unspoken for the entirety of the call, if `Negate` is set to `true`.

Amazon Transcribe and Amazon Transcribe Analytics support this rule type.

Sentiment Analysis

Alerts when a particular sentiment type is ongoing for a rolling window period. Only Transcribe Call Analytics support this rule.

Issue Detection

Alerts when an issue is detected in an utterance event. Only Transcribe Call Analytics supports this rule type.

The following example shows a real-time alert event for a `KeywordMatch` rule.

```
{
    "version": "0",
    "id": "string",
    "detail-type": "Media Insights Rules Matched",
    "source": "aws.chime",
    "account": number,
    "region": "us-east-1",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "resources": [],
    "detail":   {
        "version": "0",
        "sourceMetadata": {}
        "ruleName": "string"
        "utteranceId": "string",
        "beginTimestamp": "yyyy-mm-ddThh:mm:ssZ",
    }
}
```

Some EventBridge fields are specific to the rule type that is matched:

Keyword match fields

`utteranceId`: ID of the transcript that contains a matched keyword if you use Amazon Transcribe Call Analytics. For spoken keyword match only.

`resultId`: ID of the transcript that contains a matched keyword if you use Amazon Transcribe. For spoken keyword match only.

`beginTimestamp`: Start time of the transcript that contains a matched keyword. For spoken keyword match only.

Sentiment analysis fields

`beginTimestamp`: Start time of the rolling window for the matched sentiment.

`endTimestamp`: End time of the rolling window for the matched sentiment.

Creating an Amazon Chime SDK data lake

The Amazon Chime SDK call analytics data lake allows you to stream your machine learning powered insights and any metadata from Amazon Kinesis Data Stream to your Amazon S3 bucket. For example, using the data lake to access URLs to recordings. To create the data lake, you deploy a set of AWS CloudFormation templates from either the Amazon Chime SDK console or programmatically using the AWS CLI. The data lake enables you to query your call metadata and voice analytics data by referencing AWS Glue data tables in Amazon Athena.

Topics

Prerequisites

You must have the following items in order to create an Amazon Chime SDK lake:

- An Amazon Kinesis data stream. For more information, refer to Creating a Stream via the AWS Management Console in the *Amazon Kinesis Streams Developer Guide*.
- An S3 bucket. For more information, refer to Create your first Amazon S3 bucket in the *Amazon S3 User Guide*.

Data lake terminology and concepts

Use the following terms and concepts to understand how the data lake works.

Amazon Kinesis Data Firehose

An extract, transform, and load (ETL) service that reliably captures, transforms, and delivers streaming data to data lakes, data stores, and analytics services. For more information, see What Is Amazon Kinesis Data Firehose?

Amazon Athena

Amazon Athena is an interactive query service that enables you to analyze data in Amazon S3 using standard SQL. Athena is serverless, so you have no infrastructure to manage, and you pay only for the queries that you run. To use Athena, point to your data in Amazon S3, define the schema, and use standard SQL queries. You can also use workgroups to group users and control the resources they have access to when running queries. Workgroups enable you to manage query concurrency and prioritize query execution across different groups of users and workloads.

Glue Data Catalog

In Amazon Athena, tables and databases contain the metadata that details a schema for underlying source data. For each dataset, a table must exist in Athena. The metadata in the table tells Athena the location of your Amazon S3 bucket. It also specifies the data structure, such as column names, data types, and the table's name. Databases only hold the metadata and schema information for a dataset.

Creating multiple data lakes

Multiple data lakes can be created by providing a unique Glue database name to specify where to store call insights. For a given AWS account, there can be several call analytics configurations, each with a corresponding data lake. This means that data separation can be applied for certain use cases, such as customizing retention policy, and access policy on how the data is stored. There can be different security policies applied for access of insights, recordings, and metadata.

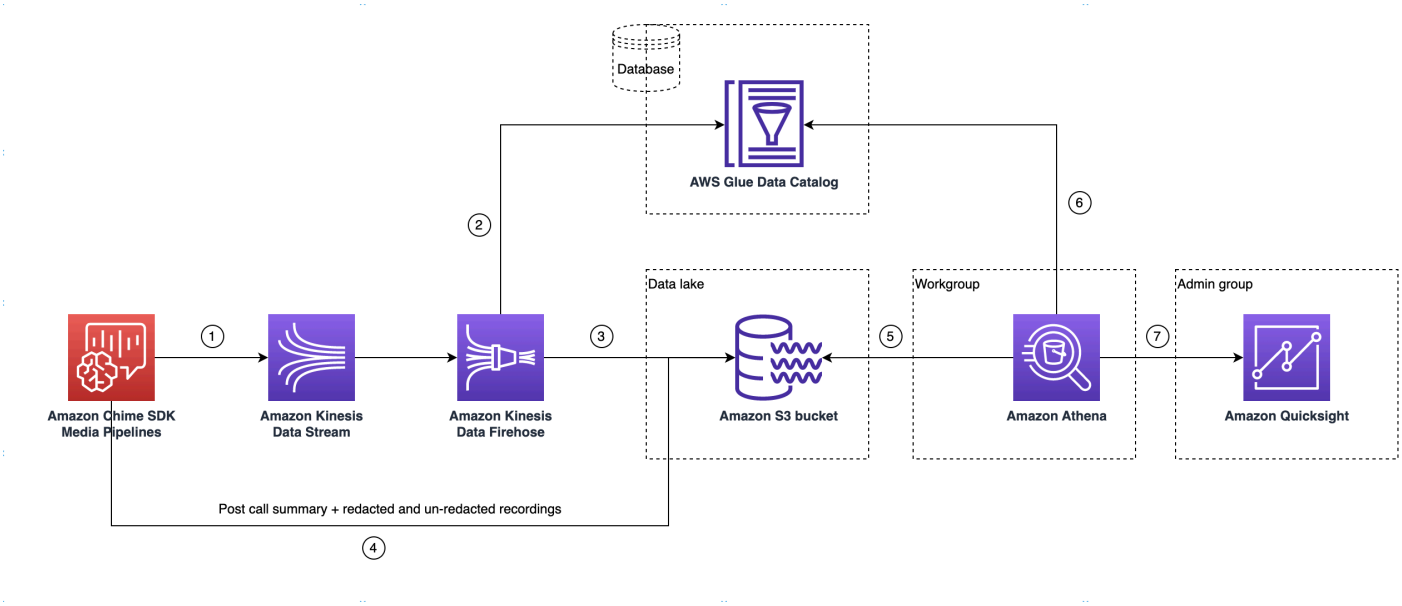
Data lake regional availability

The Amazon Chime SDK data lake is available in the following Regions.

Region	Glue table	Amazon QuickSight
us-east-1	Available	Available
us-west-2	Available	Available
eu-central-1	Available	Available

Data lake architecture

The following diagram shows the data lake architecture. Numbers in the drawing correspond to the numbered text below.



In the diagram, once you use the AWS console to deploy the CloudFormation template from the media insights pipeline configuration setup workflow, the following data flows to the Amazon S3 bucket:

1. The Amazon Chime SDK call analytics will start streaming real-time data to the customer's Kinesis Data Stream.
2. The Amazon Kinesis Firehose buffers this real-time data until it accumulates 128 MB, or 60 seconds elapses, whichever is first. Firehose then uses the `amazon_chime_sdk_call_analytics_firehose_schema` in the Glue Data Catalog to compress the data and transforms the JSON records to a parquet file.
3. The parquet file resides in your Amazon S3 bucket, in a partitioned format.
4. In addition to real-time data, post-call Amazon Transcribe Call Analytics summary .wav files (redacted and non-redacted, if specified in the configuration), and call recording .wav files are also sent to your Amazon S3 Bucket.
5. You can use Amazon Athena and standard SQL to query the data in the Amazon S3 bucket.
6. The CloudFormation template also creates a Glue Data Catalog to query this post-call summary data through Athena.

7. All the data on Amazon S3 bucket can also be visualized using Amazon QuickSight. QuickSight builds up a connection with an Amazon S3 bucket using Amazon Athena.

The Amazon Athena table uses the following features to optimize query performance:

Data partitioning

Partitioning divides your table into parts and keeps the related data together based on column values such as date, country, and region. Partitions act as virtual columns. In this case, the CloudFormation template defines partitions at table creation, which helps reduce the amount of data scanned per query and improves performance. You can also filter by partition to restrict the amount of data scanned by a query. For more information, refer to Partitioning data in Athena in the *Amazon Athena User Guide*.

This example shows partitioning structure with a date of January 1, 2023:

i.
```
s3://example-bucket/amazon_chime_sdk_data_lake
                            /serviceType=CallAnalytics/detailType={DETAIL_TYPE}/
year=2023
                            /month=01/day=01/example-file.parquet
```

ii. where `DETAIL_TYPE` is one of the following:

   a. `CallAnalyticsMetadata`
   b. `TranscribeCallAnalytics`
   c. `TranscribeCallAnalyticsCategoryEvents`
   d. `Transcribe`
   e. `Recording`
   f. `VoiceAnalyticsStatus`
   g. `SpeakerSearchStatus`
   h. `VoiceToneAnalysisStatus`

Optimize columnar data store generation

Apache Parquet uses column-wise compression, compression based on data type, and predicate pushdown to store data. Better compression ratios or skipping blocks of data means reading fewer bytes from your Amazon S3 bucket. That leads to better query performance and reduced cost. For this optimization, data conversion from JSON to parquet is enabled in the Amazon Kinesis Data Firehose.

Partition Projection

This Athena feature automatically creates partitions for each day to improve date-based query performance.

Data lake setup

Use the Amazon Chime SDK console to complete the following steps.

1. Start the Amazon Chime SDK console (https://console.aws.amazon.com/chime-sdk/home) and in the navigation pane, under **Call Analytics**, choose **Configurations**.
2. Complete Step 1, choose **Next** and on the Step 2 page, choose the **Voice Analytics** check box.
3. Under **Output details**, select the **Data warehouse to perform historical analysis** checkbox, then choose the **Deploy CloudFormation stack** link.

   The system sends you to the **Quick create stack** page in the CloudFormation console.
4. Enter a name for the stack, then enter the following parameters:
   a. `DataLakeType` – Choose **Create Call Analytics DataLake**.
   b. `KinesisDataStreamName` – Choose your stream. It should be the stream used for call analytics streaming.
   c. `S3BucketURI` – Choose your Amazon S3 bucket. The URI must have the prefix `s3://bucket-name`
   d. `GlueDatabaseName` – Choose a unique AWS Glue Database name. You cannot reuse an existing database in AWS account.
5. Choose the acknowledgment checkbox, then choose **Create data lake**. Allow 10 minutes for the system to create the lake.

Data lake setup using AWS CLI

Use AWS CLI to create a role with permissions to call CloudFormation's create stack. Follow the procedure below to create and setup the IAM roles. For more information, see Creating a stack in the *AWS CloudFormation User Guide*.

1. Create a role called *AmazonChimeSdkCallAnalytics-Datalake-Provisioning-Role* and attach a trust policy to the role allowing CloudFormation to assume the role.

   1. Create an IAM trust policy using the following template and save the file in .json format.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudformation.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
```

2. Run the **aws iam create-role** command and pass the trust policy as a parameter.

```
                                    aws iam create-role \
    --role-name AmazonChimeSdkCallAnalytics-Datalake-Provisioning-Role
    --assume-role-policy-document file://role-trust-policy.json
```

3. Note down the *role arn* that is returned from the response. *role arn* is required in the next step.

2. Create a policy with permission to create a CloudFormation stack.

   1. Create an IAM policy using the following template and save the file in .json format. This file is required when calling create-policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DeployCloudFormationStack",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

2. Run **aws iam create-policy** and pass create stack policy as a parameter.

```
                                aws iam create-policy --policy-name
 testCreateStackPolicy
--policy-document file://create-cloudformation-stack-policy.json
```

3. Note down the *role arn* that is returned from the response. *role arn* is required in the next step.

3. Attach the policy to the role **aws iam attach-role-policy**.

```
                            aws iam attach-role-policy --role-name {Role name
 created above}
--policy-arn {Policy ARN created above}
```

4. Create a CloudFormation stack and enter the required parameters: **aws cloudformation create-stack**.

   Provide parameter values for each ParameterKey using ParameterValue.

```
                            aws cloudformation create-stack  --capabilities
 CAPABILITY_NAMED_IAM
--stack-name testDeploymentStack
--template-url https://chime-sdk-assets.s3.amazonaws.com/public_templates/
AmazonChimeSDKDataLake.yaml
--parameters  ParameterKey=S3BucketURI,ParameterValue={S3 URI}
ParameterKey=DataLakeType,ParameterValue="Create call analytics datalake"
ParameterKey=KinesisDataStreamName,ParameterValue={Name of Kinesis Data Stream}
--role-arn {Role ARN created above}
```

Resources created by data lake setup

The following table lists the resources created when you create a data lake.

Resource type	Resource name and description	Service name
AWS Glue Data Catalog Database	**GlueDatabaseName** – Logically groups all AWS Glue Data tables belonging to call insights and voice analytics .	Call analytics, voice analytics
AWS Glue Data Catalog Tables	**amazon_chime_sdk_call_analytics_firehose_schema** – Combined schema for call analytics voice analytics that is fed to the Kinesis Firehose.	Call analytics, voice analytics
	call_analytics_metadata – Schema for call analytics metadata. Contains SIPmetadata and OneTimeMe tadata.	Call analytics
	call_analytics_recording_metadata – Schema for Recording and Voice Enhancement metadata	Call analytics, voice analytics
	transcribe_call_analytics – Schema for Transcrib eCallAnalytics Payload "utteranceEvent"	Call analytics
	transcribe_call_analytics_category_events – Schema for TranscribeCallAnalytics Payload "categoryEvent"	Call analytics
	transcribe_call_analytics_post_call – Schema for Post Call Transcribe Call Analytics summary payload	Call analytics
	transcribe – Schema for Transcribe Payload	Call analytics
	voice_analytics_status – Schema for voice analytics ready events	Voice analytics
	speaker_search_status – Schema for identification matches	Voice analytics
	voice_tone_analysis_status – Schema for voice tone analysis events	Voice analytics
Amazon Kinesis Data Firehose	**AmazonChimeSDK-call-analytics-*UUID*** – Kinesis Data Firehose piping data for call analytics	Call analytics, voice analytics

Resource type	Resource name and description	Service name
Amazon Athena Workgroup	**GlueDatabaseName-AmazonChimeSDKDataAnalytics** – Logical group of users to control the resources they have access to when running queries.	Call analytics, voice analytics

Configuring an Amazon QuickSight dashboard

Once you set up the data lake, you can configure an Amazon QuickSight dashboard with pre-defined metrics that visualize your data. You can use the following dashboards:

- **Transcribe Call Analytics + Voice Analytics**. Metrics include summary and detailed visuals for turn-by-turn transcripts, issues detected, outcomes, entity detection, and voice profile ID matches.
- **Transcribe + Voice Analytics**. Metrics include summary and detailed visuals for turn-by-turn transcripts, vocabulary matches, voice tone, and voice profile ID matches.

The following topics explain how to set up an Amazon QuickSight account if you don't already have one, and how to configure a dashboard.

Topics

Creating a QuickSight account

The steps in this section explain how to create an Amazon QuickSight account. If you already have an account, you can skip to Creating a QuickSight dashboard.

You can create a QuickSight account by:

- Using Amazon CloudFormation templates.
- Using the Amazon Chime SDK console.

Prerequisites

Gather the following information before you start:

- The name of your call analytics Amazon S3 bucket.
- A notification email address. The system delivers QuickSight notifications to this address.

Using CloudFormation templates to create an account

The following steps explain how to create an Amazon QuickSight account by deploying an Amazon CloudFormation template. The process only subscribes you to an Enterprise account. For information about pricing, refer to Amazon QuickSight Pricing.

To deploy the template

1. Start the AWS console and log in to your AWS account.
2. Paste the following URL into your browser's address bar. Make sure to enter your Region as indicated.

   ```
   https://region.console.aws.amazon.com/cloudformation/home?
   region=region#/stacks/quickcreate?templateURL=https://
   chime-sdk-assets.s3.amazonaws.com/public_templates/
   AmazonChimeSDKQuickSightSubscription.yaml.
   ```
3. On the **Quick create stack** page, enter the following:

   a. Under **Stack name** enter a name for your account.

   b. Under **QuickSightNotificationEmail** the email address that you gathered earlier.

   c. Under **QuickSightSubscriptionForDataVisualization**, choose **Create new AWS QuickSight account**.

   d. Under **S3BucketName**, enter the name of your Amazon S3 bucket.

   e. Select the **I acknowledge that AWS CloudFormation might create IAM resources.** checkbox.
4. Choose **Create stack**.

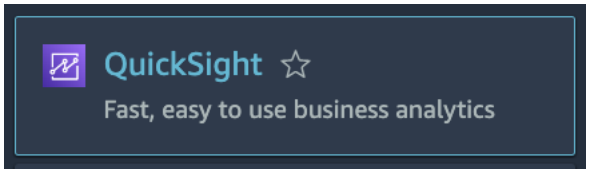
   The system takes approximately 10 minutes to create the stack.
5. When the build finishes, choose **Go to Amazon QuickSight** and enter your email address to sign in to your account.

Using the console to create an account

The following steps explain how to use the Amazon Chime SDK console to create an Amazon QuickSight account. You must use an Enterprise or Enterprise + Q account.

To use the console

1. Start the Amazon Chime SDK console at https://console.aws.amazon.com/chime-sdk/home, search for **`QuickSight`**, and in the search results choose **QuickSight**.



2. Choose **Sign up for QuickSight**.
3. Choose **Enterprise** or **Enterprise + Q**, then choose **Continue**.
4. Enter your first name, last name, phone number, and the email address that you gathered previously, then choose **Continue**.
5. Do the following:

   i. Under **Authentication method**, choose an option.

   > **Note**
   >
   > If you choose the option with federated users, you need the correct IAM permissions. For more information, refer to Signing up for an Amazon QuickSight subscription in the *Amazon QuickSight User Guide*.

   ii. Under **QuickSight Region**, select a Region.

   iii. Under **Account Info**, enter a name for the account and the email address that you gathered earlier.

   iv. Under **QuickSight access to AWS Services**, use the default role, or choose **Use an existing role** and select a role from the list.

   v. (Optional) as needed, under **Allow access and autodiscovery for these resources**, choose additional resources.

   vi. When done, choose **Finish**.

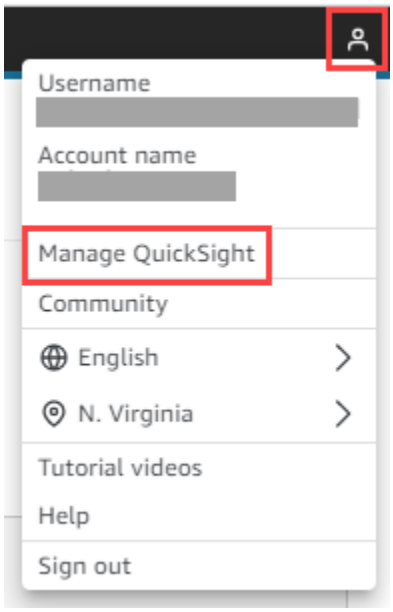
   vii. When the build finishes, choose **Go to Amazon QuickSight** and enter your email address to sign in to your account.

Configuring your QuickSight account

After you log in to your QuickSight account, you need to configure security and add yourself to a group created by the setup process.

To configure security

1. Choose the profile icon in the upper-right corner, then choose **Manage QuickSight** from the resulting menu.



2. In the navigation pane, choose **Security & permissions**.
3. Under **QuickSight access to AWS services**, choose **Manage**, and ensure the following services are selected.
   - Amazon Redshift
   - Amazon RDS
   - Amazon S3
   - Amazon Athena
   - IAM
4. Choose the **Select Amazon S3 buckets** link.
5. Select the check box next to your Amazon S3 bucket, then select the checkbox to the right, in the **Write permission for Athena Workgroup** column.
6. Choose **Finish**.
7. Choose **Save**.

To add yourself to the group

1. In the navigation pane, choose **Manage groups**, then choose the group with **Admins** in the name. For example, *S3BucketName*–**Admins**.
2. Choose **Add user**, then enter your email alias in the box that appears.

   Your name appears as **Admin –** *your-alias*.
3. Choose **Add**.

Creating a QuickSight dashboard

After you create a data lake, you can create a QuickSight dashboard that visualizes your data. You can use an Amazon CloudFormation template or the Amazon Chime SDK console to create the dashboard. The following steps explain both methods.

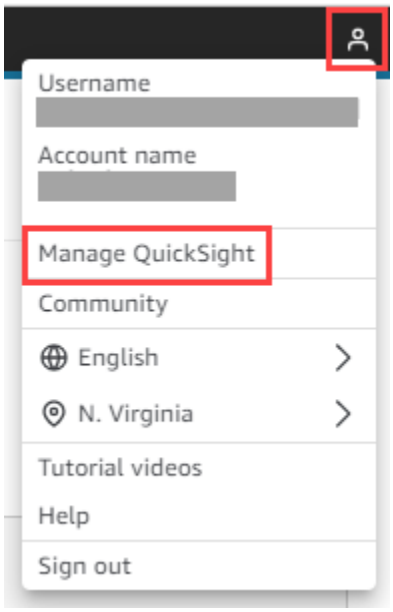
To use a template

1. Start the Amazon CloudFormation console.
2. Paste the following link into your browser's address bar:
   `https://region.console.aws.amazon.com/cloudformation/home?region=region#/stacks/quickcreate?templateURL=https://chime-sdk-assets.s3.amazonaws.com/public_templates/AmazonChimeSDKQuickSightDashboards.yaml`
3. On the **Quick create stack** page, under **Stack name**, enter a name for the account.
4. Under **ActiveQuickSightAccount**, choose **True**.
5. Under **QuicksightDashboardSelection**, choose **Call Analytics – Transcribe Call Analytics and Voice Analytics dashboard** or **Call Analytics – Transcribe and Voice Analytics dashboard**.
6. Under **Amazon S3BucketName**, enter the URI of your Amazon S3 bucket.
7. Under **GlueDatabaseName**, enter the Glue database on which you want the QuickSight dashboard deployed.
8. Choose the **I acknowledge that AWS CloudFormation might create IAM resources** check box, then choose **Create stack**.

To configure a QuickSight dashboard manually

1. Navigate to your QuickSight account.

2. In the top-right corner choose the profile icon, then choose **Manage QuickSight**.



3. In the navigation pane, choose **Manage groups**, then choose the group created by the setup process.
4. Choose **Add User**, enter your email address, then choose **Add**.

   The system takes 10 minutes to deploy the page.
5. Use the Amazon Chime SDK console to log in to your QuickSight account and use the dashboard.

Call analytics data model

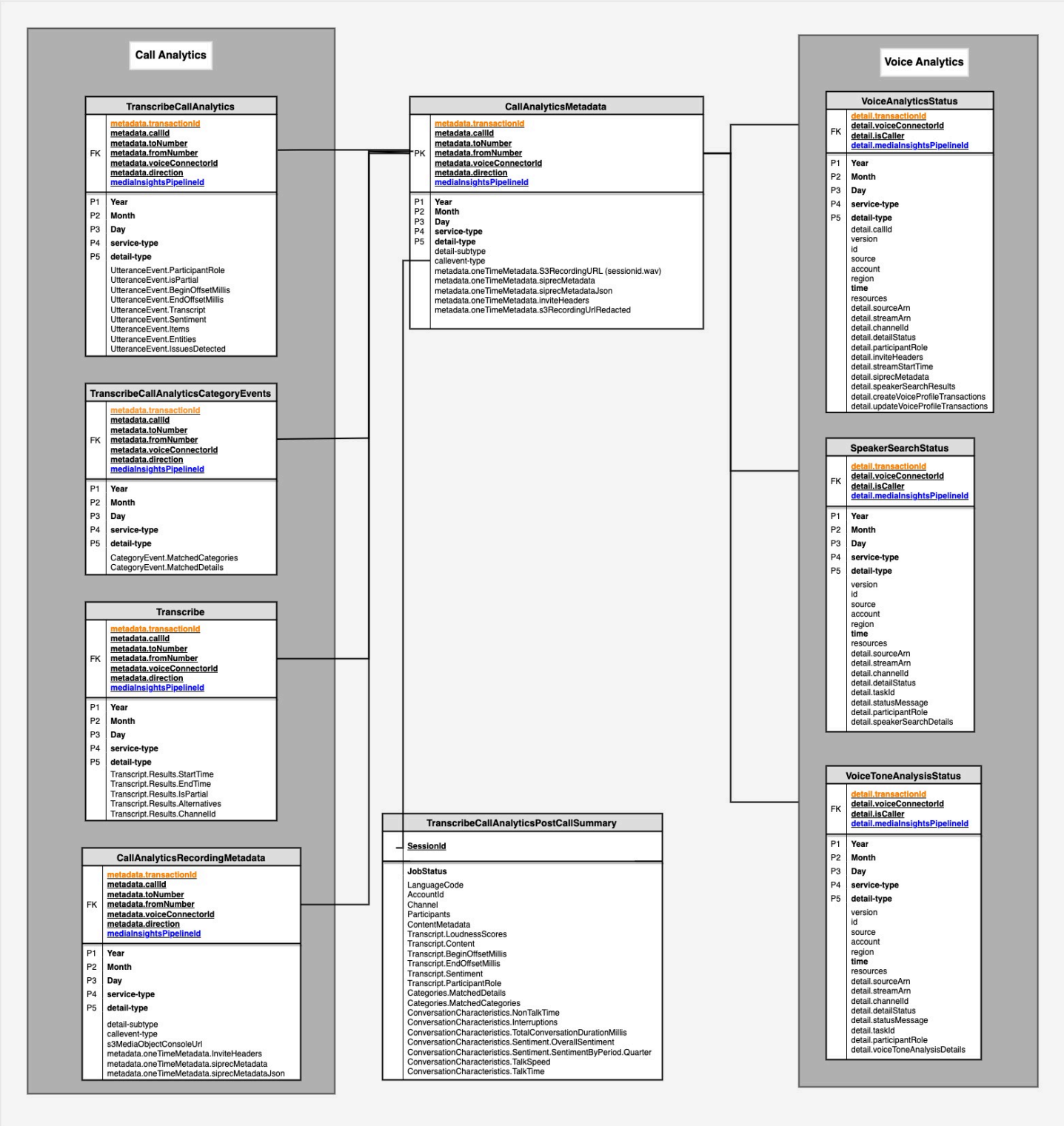
The information in this section lists and describes Amazon Chime SDK call analytics data model, a set of tables in an AWS Glue data catalog.

Topics

- Understanding the AWS Glue data catalog table structure
- Understanding the AWS Glue data catalog tables
- Extracting data in your AWS Glue data catalog for Amazon Chime SDK call analytics

Understanding the AWS Glue data catalog table structure

The following diagram shows the table structure of the AWS Glue data catalog created for Amazon Chime SDK call analytics and voice analytics sessions.



The next section lists and describes the tables and fields in the catalog.

Understanding the AWS Glue data catalog tables

The following tables list and describe the columns, data types, and elements in an Amazon Chime SDK call analytics Glue data catalog.

Topics

call_analytics_metadata

Column name	Data type	Elements	Definition
time	string		Event generation timestamp ISO 8601.
detail-type	string		Feature type related to service-type.
service-type	string		Name of the AWS service, VoiceAnal ytics or CallAnalytics.
detail-subtype	string		Used for Recording and CallAnaly ticsMetadata detail-types.

Column name	Data type	Elements	Definition
callevent-type	string		Event type associated with SIP, such as Update, Pause, Resume
mediaInsightsPipelineId	string		Amazon Chime SDK media insights pipeline ID.
metadata	string	voiceConnectorId	The Amazon Chime SDK Voice Connector ID.
		callId	The call ID of the participant for the associated usage.
		transactionId	The transaction ID of the call.
		fromNumber	E.164 origination phone number.
		toNumber	E.164 destination phone number.
		direction	Direction of the call, Outbound or Inbound.
		oneTimeMetadata.s3RecordingUrl	Amazon S3 bucket URL of the media object emitted by Transcribe Call Analytics.

Column name	Data type	Elements	Definition
		oneTimeMetadata.s3RecordingUrlRedacted	Amazon S3 bucket URL of the redacted media object emitted by Transcribe Call Analytics.
		oneTimeMetadata.siprecMetadata	SIPREC Metadata in XML format associated with the call.
		oneTimeMetadata.siprecMetadataJson	SIPREC Metadata in JSON format associated with the call.
		oneTimeMetadata.InviteHeaders	Invite headers.

call_analytics_recording_metadata

Column name	Data type	Elements	Definition
time	string		Event generation timestamp ISO 8601.
detail-type	string		Feature type related to service-type.
service-type	string		Name of the AWS service, VoiceAnalytics or CallAnalytics.
detail-subtype	string		Used for Recording and CallAnaly

Column name	Data type	Elements	Definition
			ticsMetadata detail-types.
callevent-type	string		Event type associated with SIP
mediaInsightsPipelineId	string		Amazon Chime SDK media insight pipeline ID.
s3MediaObjectConsoleUrl	string		S3 Bucket URL of the media object.
metadata	string	voiceConnectorId	The Amazon Chime SDK Voice Connector ID.
		callId	The call ID of the participant for the associated usage.
		transactionId	The transaction ID of the call.
		fromNumber	E.164 origination phone number.
		toNumber	E.164 destination phone number.
		direction	Direction of the call, Outbound or Inbound.
		voice enhancement	Feature subtype related to service-type.

Column name	Data type	Elements	Definition
		oneTimeMetadata.siprecMetadata	SIPREC Metadata in XML format associated with the call.
		oneTimeMetadata.siprecMetadataJson	SIPREC Metadata in JSON format associated with the call.
		oneTimeMetadata.InviteHeaders	Invite headers.

transcribe_call_analytics

Column name	Data type	Elements	Definition
time	string		Event generation timestamp ISO 8601.
detail-type	string		Feature type related to service-type.
service-type	string		Name of the AWS service, VoiceAnalytics or CallAnalytics.
mediaInsightsPipelineId	string		Amazon Chime SDK media insight pipeline ID.
metadata	string	voiceConnectorId	The Amazon Chime Voice Connector ID.
		callId	The call ID of the participant for the associated usage.

Column name	Data type	Elements	Definition
		transactionId	The transaction ID of the call.
		fromNumber	E.164 origination phone number.
		toNumber	E.164 destination phone number.
		direction	Direction of the call, `Outbound` or `Inbound`.
UtteranceEvent	struct	UtteranceId	The unique identifier associated with the specified `UtteranceEvent` .
		IsPartial	Indicates whether the segment in the `UtteranceEvent` is complete (FALSE) or partial (TRUE).
		ParticipantRole	Provides the role of the speaker for each audio channel, either CUSTOMER or AGENT.
		BeginOffsetMillis	The time, in milliseconds, from the beginning of the audio stream to the start of the `UtteranceEvent` .

Column name	Data type	Elements	Definition
		EndOffsetMillis	The time, in milliseconds, from the beginning of the audio stream to the start of the `UtteranceEvent` .
		Transcript	Contains transcribed text.
		Sentiment	Provides the sentiment detected in the specified segment.
		Items.beginoffsetmillis	The start time, in milliseconds, of the transcribed item.
		Items.endoffsetmillis	The end time, in milliseconds, of the transcribed item.
		Items.itemtype	The type of item identified. Options: `PRONUNCIATION` (spoken words) and `PUNCTUATION` .
		Items.content	The word or punctuation that was transcribed.

Column name	Data type	Elements	Definition
		Items.confidence	The confidence score associated with a word or phrase in your transcript. Scores are values between 0 and 1. A larger value indicates a higher probabili ty that the identifie d item correctly matches the item spoken in your media.
		Items.vocabularyfi ltermatch	Indicates whether the specified item matches a word in the vocabulary filter included in your request. If true, there is a vocabulary filter match.
		Items.stable	The partial result stabilization is enabled, Stable indicates whether the specified item is stable (true) or if it may change when the segment is complete (false).

Column name	Data type	Elements	Definition
		IssuesDetected.characteroffsets_begin	Provides the character count of the first character where a match is identified. For example, the first character associated with an issue or a category match in a segment transcript.
		IssuesDetected.characteroffsets_end	Provides the character count of the last character where a match is identified. For example, the last character associated with an issue or a category match in a segment transcript.
		Entities.beginoffsetmillis	The start time, in milliseconds, of the utterance that was identified as `PII`.
		Entities.endoffsetmillis	The end time, in milliseconds, of the utterance that was identified as `PII`.

Column name	Data type	Elements	Definition
		Entities.category	The category of information identified. The only category is `PII`.
		Entities.type	The type of PII identified. For example, `NAME` or `CREDIT_DEBIT_NUMBER`.
		Entities.content	The word or words identified as `PII`.
		Entities.confidence	The confidence score associated with the identified `PII` entity in your audio. Confidence scores range between 0 and 1. A larger value indicates a higher probability that the identified entity correctly matches the entity spoken in your media.

transcribe_call_analytics_category_events

Column name	Data type	Elements	Definition
time	string		Event generation timestamp ISO 8601.

Column name	Data type	Elements	Definition
detail-type	string		Feature type related to service-type.
service-type	string		Name of the AWS service, VoiceAnalytics or CallAnalytics.
mediaInsightsPipelineId	string		Amazon Chime SDK media insight pipeline ID.
metadata	string	voiceConnectorId	The Amazon Chime Voice Connector ID.
		callId	The call ID of the participant for the associated usage.
		transactionId	The transaction ID of the call.
		fromNumber	E.164 origination phone number.
		toNumber	E.164 destination phone number.
		direction	Direction of the call, Outbound or Inbound.
CategoryEvent	array	MatchedCategories	Lists the matches in the categories defined by the user.

transcribe_call_analytics_post_call

Column name	Data type	Elements	Definition
JobStatus	string		Event generation timestamp ISO 8601.
LanguageCode	string		Feature type related to service-type.
Transcript	struct	LoudnessScores	Measures the volume at which each participant is speaking. Use this metric to see if the caller or the agent is speaking loudly or yelling, which often indicates anger. This metric is represented as a normalized value (speech level per second of speech in a given segment) on a scale from 0 to 100, where a higher value indicates a louder voice.
		Content	Contains transcribed text.
		Id	The unique identifie r associated with the specified `UtteranceEvent` .

Column name	Data type	Elements	Definition
		BeginOffsetMillis	The time, in milliseconds, from the beginning of the audio stream to the start of the `UtteranceEvent` .
		EndOffsetMillis	The time, in milliseconds, from the beginning of the audio stream to the start of the `UtteranceEvent` .
		Sentiment	Provides the sentiment detected in the specified transcript segment.
		ParticipantRole	Provides the role of the speaker for each audio channel, either `CUSTOMER` or `AGENT`.
		IssuesDetected.CharacterOffsets.Begin	Provides the character offset to the first character where a match is identified. For example, the first character associated with an issue in a transcript segment.

Column name	Data type	Elements	Definition
		IssuesDetected.CharacterOffsets.End	Provides the character offset to the last character where a match is identified. For example, the last character associated with an issue in a transcript segment.
		OutcomesDetected.CharacterOffsets.Begin	Provides the outcome, or resolution, identified in the call.
		OutcomesDetected.CharacterOffsets.End	
		ActionItemsDetected.CharacterOffsets.Begin	Lists any action items identified in the call.
		ActionItemsDetected.CharacterOffsets.End	
AccountId	string		The AWS account Id
Categories	struct	MatchedCategories	Lists the matched categories.
		MatchedDetails	Lists the time, in milliseconds, from the beginning of the audio stream to when the Match in the category was detected.

Column name	Data type	Elements	Definition
Channel	string	Channel	Indicates a Voice channel.
Participants	array	ParticipantRole	Provides the role of the speaker for each audio channel, `CUSTOMER` or `AGENT`.
ConversationCharacteristics	struct	NonTalkTime	Measures periods of time that do not contain speech. Use this metric to find long periods of silence, such as a customer on hold for an excessive amount of time.
		Interruptions	Measures if and when one participant cuts off the other participant mid-sentence. Frequent interruptions may be associated with rudeness or anger, and could correlate to negative sentiment for one or both participants.
		TotalConversationDurationMillis	Total length of the conversation.

Column name	Data type	Elements	Definition
		Sentiment.OverallSentiment.AGENT	`OverallSentiment` label for the Agent.
		Sentiment.OverallSentiment.CUSTOMER	`OverallSentiment` label for the `Customer`.
		Sentiment.SentimentByPeriod.QUARTER.AGENT	Sentiment labels for each quarter for the `Agent`.
		Sentiment.SentimentByPeriod.QUARTER.CUSTOMER	Sentiment labels for each quarter for the `Customer`.
		TalkSpeed	Measures the speed at which both participants are speaking. Comprehension can be affected if one participant speaks too quickly. This metric is measured in words per minute.

Column name	Data type	Elements	Definition
		TalkTime	Measures the amount of time (in milliseco nds) each participa nt spoke during the call. Use this metric to help identify if one participant is dominating the call or if the dialogue is balanced.
SessionId	string		`SessionId` for the call
ContentMetadata	string		Field that labels raw vs. redacted content per the customer specified configura tion.

transcribe

Column name	Data type	Elements	Definition
time	string		Event generation timestamp ISO 8601.
detail-type	string		Feature type related to service-type.
service-type	string		Name of the AWS service, VoiceAnal ytics or CallAnalytics.

Column name	Data type	Elements	Definition
mediaInsightsPipelineId	string		Amazon Chime SDK media insight pipeline ID.
metadata	string	voiceConnectorId	The Amazon Chime Voice Connector ID.
		callId	The call ID of the participant for the associated usage.
		transactionId	The transaction ID of the call.
		fromNumber	E.164 origination phone number.
		toNumber	E.164 destination phone number.
		direction	Direction of the call, `Outbound` or `Inbound`.
TranscriptEvent	struct	ResultId	The unique identifier for the `Result`.
		StartTime	The start time, in milliseconds, of the `Result`.
		EndTime	The end time, in milliseconds, of the `Result`.

Column name	Data type	Elements	Definition
		IsPartial	Indicates whether the segment is complete. If `IsPartial` is `true`, the segment is not complete. Otherwise, the segment is complete.
		ChannelId	The ID of the channel associated with the audio stream.
		Alternatives.Entities	Contains entities identified as personally identifia ble information (PII) in your transcription output.
		Alternatives.Items .Confidence	The confidence score associated with a word or phrase in your transcript. Confidence scores are values between 0 and 1. A larger value indicates a higher probability that the identified item correctly matches the item spoken in your media.
		Alternatives.Items .Content	The transcribed word or punctuation mark.

Column name	Data type	Elements	Definition
		Alternatives.Items .EndTime	The end time, in milliseconds, of the transcribed item.
		Alternatives.Items .Speaker	If speaker partition ing is enabled, `Speaker` labels the speaker of the specified item.
		Alternatives.Items .Stable	If partial result stabilization is enabled. `Stable` indicates whether the specified item is stable (true) or if it may change when the segment is complete (false).
		Alternatives.Items .StartTime	The start time, in milliseconds, of the transcribed item.
		Alternatives.Items .Type	The type of item identified. Options: `PRONUNCIATION` (spoken words) and `PUNCTUATION` .

Column name	Data type	Elements	Definition
		Alternatives.Items.VocabularyFilterMatch	Indicates whether the specified item matches a word in the vocabulary filter included in your request. If true, there is a vocabulary filter match.
		Alternatives.Transcript	Contains transcribed text.

voice_analytics_status

Column name	Data type	Elements	Definition
time	string		Event generation timestamp ISO 8601.
detail-type	string		Feature type related to service-type.
service-type	string		Name of the AWS service, VoiceAnalytics or CallAnalytics.
source	string		AWS service that produces the event.
account	string		AWS Account ID.
region	string		AWS Account Region.
version	string		Version of the event schema.

Column name	Data type	Elements	Definition
id	string		Unique ID of the event
detail	struct	taskId	Unique ID of the task.
		isCaller	Indicates whether the participant is caller or not.
		streamStartTime	Start time of the stream.
		transactionId	The transaction ID of the call.
		voiceConnectorId	The Amazon Chime Voice Connector ID.
		callId	The call ID of the participant for the associated usage.
		detailStatus	Detailed feature type related to service-t ype.
		statusMessage	Status of task ID success or failure.
		mediaInsightsPipel ineId	Amazon Chime SDK media insight pipeline ID. This field is populated only for speaker search tasks started through the Media Pipelines SDK, not the Voice SDK.

Column name	Data type	Elements	Definition
		sourceArn	The resource ARN for which the task is run on
		streamArn	The Kinesis Video Stream ARN that the task is run for. This field is populated only for speaker search tasks started through the Media Pipelines SDK, not the Voice SDK.
		channelId	The channel of the streamArn that the task is run for. This field is populated only for speaker search tasks started through the Media Pipelines SDK, not the Voice SDK.
		speakerSearchDetails.voiceProfileId	ID of a voice profile enrolled whose voice embedding matches closely with the speaker in the call.

Column name	Data type	Elements	Definition
		speakerSearchDetails.confidenceScore	Number between [0, 1] where a larger number means the machine learning model is more confident about the voice profile match.

speaker_search_status

Column name	Data type	Elements	Definition
time	string		Event generation timestamp ISO 8601.
detail-type	string		Feature type related to service-type.
service-type	string		Name of the AWS service, VoiceAnalytics or CallAnalytics.
source	string		AWS service that produces the event.
account	string		AWS Account ID.
region	string		AWS Account Region.
version	string		Version of the event schema.
id	string		Unique ID of the event
detail	struct	taskId	Unique ID of the task.

Column name	Data type	Elements	Definition
		isCaller	Indicates whether the participant is caller or not.
		transactionId	The transaction ID of the call. This field is populated if the task originates from a call made through a Voice Connector.
		voiceConnectorId	The Amazon Chime Voice Connector ID. This field is populated if the task originates from a call made through a Voice Connector.
		mediaInsightsPipelineId	The media insights pipeline ID. This field is populated only for speaker search tasks started through the Media Pipelines SDK, not the Voice SDK.
		sourceArn	The resource ARN for which the task is run on.

Column name	Data type	Elements	Definition
		streamArn	The Kinesis Video Stream ARN that the task is run for. This field is populated only for speaker search tasks started through the Media Pipelines SDK, not the Voice SDK.
		channelId	The channel of the streamArn that the task is run for. This field is populated only for speaker search tasks started through the Media Pipelines SDK, not the Voice SDK.
		participantRole	The participant role associated with the channelId in the streamArn. This field is populated only for speaker search tasks started through the Media Pipelines SDK, not the Voice SDK.
		detailStatus	Detailed feature type related to service-t ype.

Column name	Data type	Elements	Definition
		statusMessage	Status of task ID, success or failure.
		speakerSearchDetails.voiceProfileId	ID of a voice profile enrolled whose voice embedding matches closely with the speaker in the call.
		speakerSearchDetails.confidenceScore	Number between [0, 1] where a larger number means the machine learning model is more confident about the voice profile match.

voice_tone_analysis_status

Column name	Data type	Elements	Definition
time	string		Event generation timestamp ISO 8601.
detail-type	string		Feature type related to service-type.
service-type	string		Name of the AWS service, VoiceAnalytics or CallAnalytics.
source	string		AWS service that produces the event.
account	string		AWS Account ID.

Column name	Data type	Elements	Definition
region	string		AWS Account Region.
version	string		Version of the event schema.
id	string		Unique ID of the event
detail	struct	taskId	Unique ID of the task.
		isCaller	Indicates whether the participant is caller or not.
		transactionId	The transaction ID of the call. This field is populated if the task originates from a call made through a Voice Connector.
		voiceConnectorId	The Amazon Chime Voice Connector ID. This field is populated if the task originates from a call made through a Voice Connector.
		mediaInsightsPipelineId	The media insights pipeline ID. This field is populated only for speaker search tasks started through the Media Pipelines SDK, not the Voice SDK.

Column name	Data type	Elements	Definition
		sourceArn	The resource ARN for which the task is run on.
		streamArn	The Kinesis Video Stream ARN that the task is run for. This field is populated only for speaker search tasks started through the Media Pipelines SDK, not the Voice SDK.
		channelId	The channel of the streamArn that the task is run for. This field is populated only for speaker search tasks started through the Media Pipelines SDK, not the Voice SDK.
		participantRole	The participant role associated with the channelId in the streamArn. This field is populated only for speaker search tasks started through the Media Pipelines SDK, not the Voice SDK.

Column name	Data type	Elements	Definition
		statusMessage	Status of task ID success or failure.
		voiceToneAnalysisDetails.startFragmentNumber	Starting fragment number associated with the streamArn.
		voiceToneAnalysisDetails.currentAverageVoiceTone.startTime	Starting timestamp in ISO8601 format for the speaker's call audio that the current average sentiment is based on.
		voiceToneAnalysisDetails.currentAverageVoiceTone.endTime	Ending timestamp in ISO8601 format for the speaker's call audio that the current average sentiment is based on.
		voiceToneAnalysisDetails.currentAverageVoiceTone.beginOffsetMillis	Beginning offset in milliseconds from the starting fragment for the speaker's call audio that the current average sentiment is based on.

Column name	Data type	Elements	Definition
		voiceToneAnalysisDetails.currentAverageVoiceTone.endOffsetMillis	Ending offset in milliseconds from the starting fragment for the speaker's call audio that the current average sentiment is based on.
		voiceToneAnalysisDetails.currentAverageVoiceTone.voiceToneScore.positive	Probabilistic likelihood between [0, 1] that the speaker's sentiment is positive.
		voiceToneAnalysisDetails.currentAverageVoiceTone.voiceToneScore.negative	Probabilistic likelihood between [0, 1] that the speaker's sentiment is negative.
		voiceToneAnalysisDetails.currentAverageVoiceTone.voiceToneScore.neutral	Probabilistic likelihood between [0, 1] that the speaker's sentiment is neutral.
		voiceToneAnalysisDetails.currentAverageVoiceTone.voiceToneLabel	Label with highest probability for the average voice tone score.
		voiceToneAnalysisDetails.overallAverageVoiceTone.startTime	Starting timestamp in ISO8601 format for the speaker's call audio that the overall average sentiment is based on.

Column name	Data type	Elements	Definition
		voiceToneAnalysisDetails.overallAverageVoiceTone.endTime	Ending timestamp in ISO8601 format for the speaker's call audio that the overall average sentiment is based on.
		voiceToneAnalysisDetails.overallAverageVoiceTone.beginOffsetMillis	Beginning offset in milliseconds from the starting fragment for the speaker's call audio that the overall average sentiment is based on.
		voiceToneAnalysisDetails.overallAverageVoiceTone.endOffsetMillis	Ending offset in milliseconds from the starting fragment for the speaker's call audio that the overall average sentiment is based on.
		voiceToneAnalysisDetails.overallAverageVoiceTone.voiceToneScore.positive	Probabilistic likelihood between [0, 1] that the speaker's sentiment is positive.
		voiceToneAnalysisDetails.overallAverageVoiceTone.voiceToneScore.negative	Probabilistic likelihood between [0, 1] that the speaker's sentiment is negative.

Column name	Data type	Elements	Definition
		voiceToneAnalysisDetails.overallAverageVoiceTone.voiceToneScore.neutral	Probabilistic likelihood between [0, 1] that the speaker's sentiment is neutral.
		voiceToneAnalysisDetails.overallAverageVoiceTone.voiceToneLabel	Sentiment label (positive, negative, or neutral) with the highest sentiment score.

Extracting data in your AWS Glue data catalog for Amazon Chime SDK call analytics

Use these sample queries to extract and organize the data in your Amazon Chime SDK call analytics Glue data catalog.

> **Note**
>
> For information about connecting to Amazon Athena and querying your Glue data catalog, see Connecting to Amazon Athena with ODBC.

Expand each section as needed.

Extracting values from metadata (STRING datatype) in the call_analytics_metadata table

`call_analytics_metadata` has the `metadata` field in a JSON string format. Use the json_extract_scalar function in Athena to query the elements in this string.

```
SELECT
    json_extract_scalar(metadata,'$.voiceConnectorId') AS "VoiceConnector ID",
    json_extract_scalar(metadata,'$.fromNumber') AS "From Number",
    json_extract_scalar(metadata,'$.toNumber') AS "To Number",
    json_extract_scalar(metadata,'$.callId') AS "Call ID",
    json_extract_scalar(metadata,'$.direction') AS Direction,
```

```
    json_extract_scalar(metadata,'$.transactionId') AS "Transaction ID"
FROM
    "GlueDatabaseName"."call_analytics_metadata"
```

Querying SIPRECMetadata updates in the call_analytics_metadata table

The `call_analytics_metadata` field has the metadata field in a JSON string format. `metadata` has another nested object called `oneTimeMetadata`, this object contains SIPRec Metadata in original XML and transformed JSON formats. Use the `json_extract_scalar` function in Athena to query the elements in this string.

```
SELECT
    json_extract_scalar(metadata,'$.voiceConnectorId') AS "VoiceConnector ID",
    json_extract_scalar(metadata,'$.fromNumber') AS "From Number",
    json_extract_scalar(metadata,'$.toNumber') AS "To Number",
    json_extract_scalar(metadata,'$.callId') AS "Call ID",
    json_extract_scalar(metadata,'$.direction') AS Direction,
    json_extract_scalar(metadata,'$.transactionId') AS "Transaction ID",

 json_extract_scalar(json_extract_scalar(metadata,'$.oneTimeMetadata'),'$.siprecMetadata')
 AS "siprec Metadata XML",

 json_extract_scalar(json_extract_scalar(metadata,'$.oneTimeMetadata'),'$.siprecMetadataJson')
 AS "Siprec Metadata JSON",

 json_extract_scalar(json_extract_scalar(metadata,'$.oneTimeMetadata'),'$.inviteHeaders')
 AS "Invite Headers"
FROM
    "GlueDatabaseName"."call_analytics_metadata"
WHERE
    callevent-type = "update";
```

Extracting values from metadata (STRING datatype) in the call_analytics_recording_metadata table

`call_analytics_recording_metadata` has the metadata field in a JSON string format. Use the [json_extract_scalar function](#) in Athena to query the elements in this string.

```
SELECT
    json_extract_scalar(metadata,'$.voiceConnectorId') AS "VoiceConnector ID",
    json_extract_scalar(metadata,'$.fromNumber') AS "From Number",
    json_extract_scalar(metadata,'$.toNumber') AS "To Number",
```

```
    json_extract_scalar(metadata,'$.callId') AS "Call ID",
    json_extract_scalar(metadata,'$.direction') AS Direction,
    json_extract_scalar(metadata,'$.transactionId') AS "Transaction ID"
FROM
    "GlueDatabaseName"."call_analytics_recording_metadata"
WHERE
    detail-subtype = "Recording"
```

Extracting values from detail (STRUCT datatype) in the voice_analytics_status table

`voice_analytics_status` has a details field in the `struct` data type. The following example shows how to query a `struct` data type field:

```
SELECT
    detail.transactionId AS "Transaction ID",
    detail.voiceConnectorId AS "VoiceConnector ID",
    detail.siprecmetadata AS "Siprec Metadata",
    detail.inviteheaders AS "Invite Headers",
    detail.streamStartTime AS "Stream Start Time"
FROM
    "GlueDatabaseName"."voice_analytics_status"
```

Joining the voice_analytics_status and call_analytics_metadata tables

The following example query joins `call_analytics_metadata` and `voice_analytics_status`:

```
SELECT
    a.detail.transactionId AS "Transaction ID",
    a.detail.voiceConnectorId AS "VoiceConnector ID",
    a.detail.siprecmetadata AS "Siprec Metadata",
    a.detail.inviteheaders AS "Invite Headers",
    a.detail.streamStartTime AS "Stream Start Time"
    json_extract_scalar(b.metadata,'$.fromNumber') AS "From Number",
    json_extract_scalar(b.metadata,'$.toNumber') AS "To Number",
    json_extract_scalar(b.metadata,'$.callId') AS "Call ID",
    json_extract_scalar(b.metadata,'$.direction') AS Direction
FROM
    "GlueDatabaseName"."voice_analytics_status" a
INNER JOIN
    "GlueDatabaseName"."call_analytics_metadata" b
ON a.detail.transactionId = json_extract_scalar(b.metadata,'$.transactionId')
```

Extracting transcripts from the transcribe_call_analytics_post_call table

transcribe_call_analytics_post_call has transcript field in struct format with nested arrays. Use the following query to un-nest the arrays:

```
SELECT
    jobstatus,
    languagecode,
    IF(CARDINALITY(m.transcript)=0 OR CARDINALITY(m.transcript) IS NULL, NULL,
 e.transcript.id) AS utteranceId,
    IF(CARDINALITY(m.transcript)=0 OR CARDINALITY(m.transcript) IS NULL, NULL,
 e.transcript.content) AS transcript,
    accountid,
    channel,
    sessionid,
    contentmetadata.output AS "Redaction"
FROM
    "GlueDatabaseName"."transcribe_call_analytics_post_call" m
CROSS JOIN UNNEST
    (IF(CARDINALITY(m.transcript)=0, ARRAY[NULL], transcript)) AS e(transcript)
```

Joining the transcribe_call_analytics_post_call and call_analytics_metadata tables

The following query joins transcribe_call_analytics_post_call and call_analytics_metadata:

```
WITH metadata AS(
  SELECT
    from_iso8601_timestamp(time) AS "Timestamp",
    date_parse(date_format(from_iso8601_timestamp(time), '%m/%d/%Y %H:%i:%s') , '%m/%d/
%Y %H:%i:%s') AS "DateTime",
    date_parse(date_format(from_iso8601_timestamp(time) , '%m/%d/%Y') , '%m/%d/%Y') AS
 "Date",
    date_format(from_iso8601_timestamp(time) , '%H:%i:%s')  AS "Time",
    mediainsightspipelineid,
    json_extract_scalar(metadata,'$.toNumber') AS "To Number",
    json_extract_scalar(metadata,'$.voiceConnectorId') AS "VoiceConnector ID",
    json_extract_scalar(metadata,'$.fromNumber') AS "From Number",
    json_extract_scalar(metadata,'$.callId') AS "Call ID",
    json_extract_scalar(metadata,'$.direction') AS Direction,
    json_extract_scalar(metadata,'$.transactionId') AS "Transaction ID",

 REGEXP_REPLACE(REGEXP_EXTRACT(json_extract_scalar(metadata,'$.oneTimeMetadata.s3RecordingUrl')
 '[^/]+(?=\.[^.]+$)'), '\.wav$', '') AS "SessionID"
```

```
  FROM
    "GlueDatabaseName"."call_analytics_metadata"
),
transcript_events AS(
  SELECT
    jobstatus,
    languagecode,
    IF(CARDINALITY(m.transcript)=0 OR CARDINALITY(m.transcript) IS NULL, NULL,
 e.transcript.id) AS utteranceId,
    IF(CARDINALITY(m.transcript)=0 OR CARDINALITY(m.transcript) IS NULL, NULL,
 e.transcript.content) AS transcript,
    accountid,
    channel,
    sessionid,
    contentmetadata.output AS "Redaction"
  FROM
    "GlueDatabaseName"."transcribe_call_analytics_post_call" m
  CROSS JOIN UNNEST
    (IF(CARDINALITY(m.transcript)=0, ARRAY[NULL], transcript)) AS e(transcript)
)
SELECT
    jobstatus,
    languagecode,
    a.utteranceId,
    transcript,
    accountid,
    channel,
    a.sessionid,
    "Redaction"
    "Timestamp",
    "DateTime",
    "Date",
    "Time",
    mediainsightspipelineid,
    "To Number",
    "VoiceConnector ID",
    "From Number",
    "Call ID",
    Direction,
    "Transaction ID"
FROM
    "GlueDatabaseName"."transcribe_call_analytics_post_call" a
LEFT JOIN
    metadata b
```

```
ON
    a.sessionid = b.SessionID
```

Querying media object URLs for Voice enhancement call recording

The following example query joins `Voice enhancement call recording` URL:

```
SELECT
    json_extract_scalar(metadata,'$.voiceConnectorId') AS "VoiceConnector ID",
    json_extract_scalar(metadata,'$.fromNumber') AS "From Number",
    json_extract_scalar(metadata,'$.toNumber') AS "To Number",
    json_extract_scalar(metadata,'$.callId') AS "Call ID",
    json_extract_scalar(metadata,'$.direction') AS Direction,
    json_extract_scalar(metadata,'$.transactionId') AS "Transaction ID",
    s3MediaObjectConsoleUrl
FROM
    {GlueDatabaseName}."call_analytics_recording_metadata"
WHERE
    detail-subtype = "VoiceEnhancement"
```

Using Amazon Chime SDK voice analytics

The Amazon Chime SDK voice analytics feature enables you to implement speaker search and voice tone analysis. You use speaker search to identify and enroll new callers, and to identify repeat callers and assign a confidence score to those identifications. You use voice tone analysis to predict a caller's sentiment as `negative`, `neutral`, or `positive`.

You run voice analytics as an optional component of an Amazon Chime SDK call analytics session.

Voice analytics works with media insights pipelines or Amazon Chime SDK Voice Connectors calls. We recommend using the Media Pipelines SDK and invoking tasks on a media insights pipeline for finer grained control over, and information about, the tasks.

You can use Voice Connectors to ensure backward compatibility, but we only update the media insights pipeline APIs with new features.

For more information about creating and using Voice Connectors, see Managing Amazon Chime SDK Voice Connectors in the *Amazon Chime SDK Administrator Guide*.

Voice analytics also provides:

- Asynchronous task processing. Tasks run independently from each other.
- Control over when you process insights.

You can initiate voice analytics by calling the StartSpeakerSearchTask and StartVoiceToneAnalysisTask APIs.

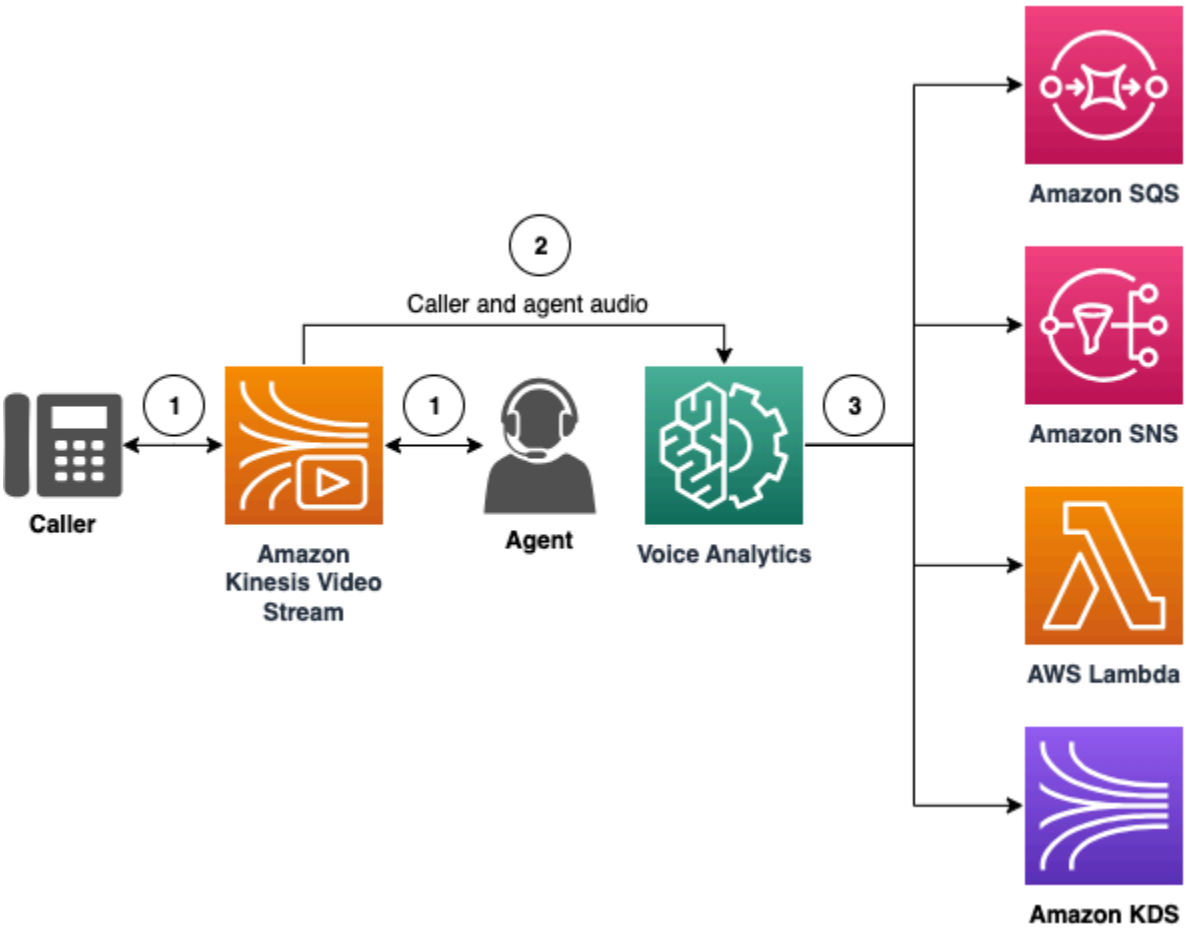
The following topics explain how to use voice analytics.

Topics

Understanding voice analytics architecture

The topics in this section provide an overview of the Amazon Chime SDK voice analytics architecture, including the data flows for each feature.

This diagram provides a high-level view of how data flows through voice analytics.



In the diagram:

1. Audio is streamed to a Kinesis Video Stream for a caller and agent. You can use a Kinesis Video Streams producer or Amazon Chime SDK Voice Connector streaming to do this. For more information, see Understanding workflows for machine-learning based analytics in this guide, and Streaming Amazon Chime SDK Voice Connector media to Kinesis in the *Amazon Chime SDK Administrator Guide*.
2. An application or builder triggers speaker search, voice tone analysis, or both, for the audio stream after the caller consents.
3. During the call, voice analytics sends notifications to a target, either Amazon Simple Queue Service (SQS), Amazon Simple Notification Service (SNS), AWS Lambda, or Amazon Kinesis Data Streams.

In addition, voice analytics provides these tools for managing the data that it generates.

Voice profiles

The combination of a voice embedding, the embedding's unique ID, and its expiration date. Voice profiles expire after three years for security reasons, and because voices change over time. To avoid re-creating voice profiles, call the UpdateVoiceProfile API. For more information about expiration dates, see Understanding data retention for Amazon Chime SDK voice analytics.

To enroll a voice embedding, or to update an enrolled voice embedding, you must call the CreateVoiceProfile or UpdateVoiceProfile APIs within 24 hours after the call ends.

Voice profile domains

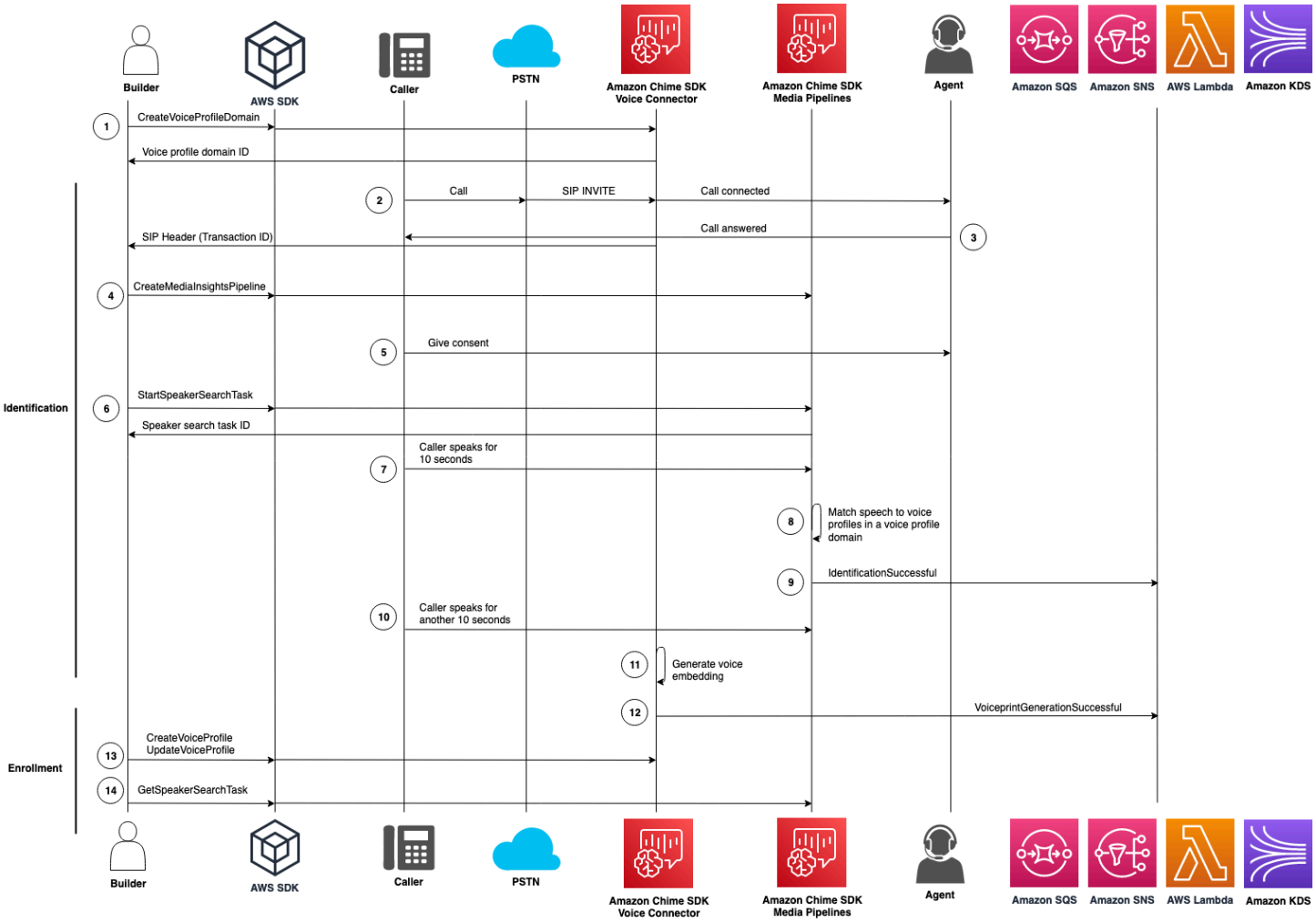
A collection of voice profiles.

Sample speaker search workflow

> **Important**
>
> The speaker search function involves the creation of a voice embedding, which can be used compare the voice of a caller against previously stored voice data. The collection, use, storage, and retention of biometric identifiers and biometric information in the form of a digital voiceprint may require the caller's informed consent via a written release. Such consent is required under various state laws, including biometrics laws in Illinois, Texas, Washington and other state privacy laws. Before using the speaker search feature, you must provide all notices, and obtain all consents as required by applicable law, and under the AWS service terms governing your use of the feature.

The following diagram shows an example data flow through a speaker search analysis task. Numbered text below the image describe each step of the process.

> **Note**
>
> The diagram assumes you have already configured an Amazon Chime SDK Voice Connector with a call analytics configuration that has a `VoiceAnalyticsProcessor`. For more information, see Recording Voice Connector calls.



In the diagram:

1. You or a system administrator create a voice profile domain for storing voice embeddings and voice profiles. For more information about creating voice profile domains, see Creating voice profile domains, in the *Amazon Chime SDK Administrator Guide*. You can also use the CreateVoiceProfileDomain API.
2. A caller dials in using a phone number assigned to an Amazon Chime SDK Voice Connector. Or, an agent uses a Voice Connector number to make an outbound call.
3. The Amazon Chime SDK Voice Connector service creates a transaction ID and associates it with the call.
4. Assuming your application subscribes to EventBridge events, your application calls the CreateMediaInsightsPipeline API with the with the media insights pipeline configuration and Kinesis Video Stream ARNs for the Voice Connector call.

For more information about using EventBridge, refer to Understanding workflows for machine-learning based analytics.

5. Your application—such as an Interactive Voice Response system—or agent provides notice to the caller regarding call recording and the use of voice embeddings for voice analytics and seeks their consent to participate.
6. Once the caller provides consent, your application or agent can call the StartSpeakerSearchTask API through the Voice SDK if you have a Voice Connector and a transaction ID. Or, if you have a media insights pipeline ID instead of a transaction ID, you call the StartSpeakerSearchTask API in the Media pipelines SDK.

   Once the caller provides consent, your application or agent calls the `StartSpeakerSearchTask` API. You must pass the Voice Connector ID, transaction ID, and voice profile domain ID to the API. A speaker search task ID is returned to identify the asynchronous task.

   > **Note**
   >
   > Before invoking the `StartSpeakerSearchTask` API in either of the SDKs, you must provide any necessary notices, and obtain any necessary consents, as required by law and under the AWS service terms.

7. The system accumulates 10 seconds of the caller's voice. The caller must speak for at least that amount of time. The system doesn't capture or analyze silence.
8. The media insights pipeline compares the speech to the voice profiles in the domain and lists top 10 high confidence matches. If it doesn't find a match, the Voice Connector creates a voice profile.
9. The media insights pipeline service sends a notification event to the configured notification targets.
10. The caller continues speaking and provides an additional 10 seconds of non-silence speech.
11. The media insights pipeline generates an enrollment voice embedding that you can use to create a voice profile or update an existing voice profile.
12. The media insights pipeline sends a `VoiceprintGenerationSuccessful` notification to the configured notification targets.
13. Your application calls the CreateVoiceProfile or UpdateVoiceProfile APIs to create or update the profile.

14. Your application calls the GetSpeakerSearchTask API as needed to get the latest status of the speaker search task.

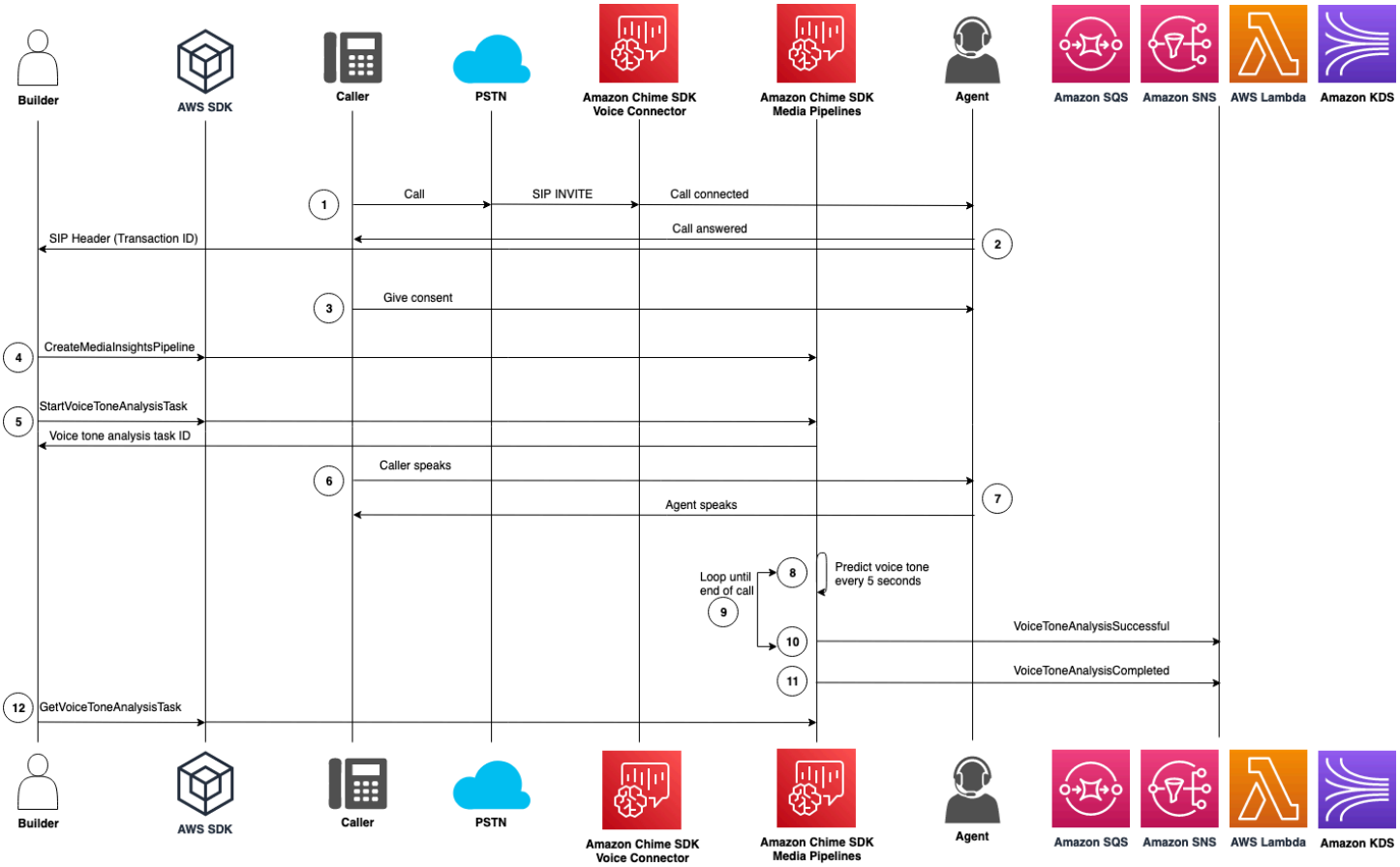
Sample voice tone analysis workflow

> **Important**
>
> Voice tone analysis involves making predictions on a speaker's sentiment based on linguistic and tonal information. You must not use sentiment analysis in any manner prohibited by law, including in relation to making decisions about an individual that would produce legal or similarly significant impacts on such individuals (e.g., related to employment, housing, credit worthiness, or financial offers, etc.).

Voice tone analysis analyzes the voices of the people on a call and predicts their sentiment, either `positive`, `negative`, or `neutral`.

The following diagram shows an example workflow for a voice tone analysis. Numbered items below the image describe each step of the process.

> **Note**
>
> The diagram assumes you have already configured an Amazon Chime SDK Voice Connector with a call analytics configuration that has a `VoiceAnalyticsProcessor`. For more information, see Recording Voice Connector calls.



In the diagram:

1. A caller dials in using a phone number assigned to an Amazon Chime SDK Voice Connector. Or, an agent uses a Voice Connector number to make an outbound call.
2. The Voice Connector service creates a transaction ID and associates it with the call.
3. Your application—such as an Interactive Voice Response system—or agent provides notice to the caller regarding call recording and the use of voice embeddings for voice analytics and seeks their consent to participate.
4. Assuming your application subscribes to EventBridge events, your application calls the CreateMediaInsightsPipeline API with the with the media insights pipeline configuration and Kinesis Video Stream ARNs for the Voice Connector call.

   For more information about using EventBridge, refer to Understanding workflows for machine-learning based analytics.
5. Once the caller provides consent, your application or agent can call the StartSpeakerSearchTask API through the Voice SDK if you have a Voice Connector and a transaction ID. Or, if you have a

media insights pipeline ID instead of a transaction ID, you call the StartSpeakerSearchTask API in the Media pipelines SDK.

Once the caller provides consent, your application or agent calls the `StartSpeakerSearchTask` API. You must pass the Voice Connector ID, transaction ID, and voice profile domain ID to the API. A speaker search task ID is returned to identify the asynchronous task.

6. The user speaks throughout the call.
7. The agent speaks throughout the call.
8. Every 5 seconds, the media insights pipeline uses a machine learning model to analyze the last 30 seconds of speech and predict the caller's tone for that interval, and for the entire call from the time when `StartVoiceToneAnalysisTask` was first called.
9. The media insights pipeline sends a notification with that information to the configured notification targets. You can identify the notification based on its stream ARN and channel ID. For more information, refer to Understanding notifications, later in this section.
10. Repeat steps 9 and 10 until the call ends.
11. At the end of the call, the media insights pipeline sends one final notification with the current average tone prediction for the last 30 seconds, plus the average tone of the entire call.
12. Your application calls the GetVoiceToneAnalysisTask API as needed to get the latest status of the voice tone analysis task.

> **Note**
>
> The `GetVoiceToneAnalysisTask` API doesn't stream the tone data.

> **Note**
>
> The GetVoiceToneAnalysisTask API doesn't return voice tone data.

Polling for task results

> **Important**
>
> By default, voice analytics makes results available for 7 days, then it deletes the data automatically. You must store your task data if you want to use it for a longer time, or to comply with data-retention laws. For more information, see Understanding data retention for Amazon Chime SDK voice analytics, later in this guide.

Voice analytics tries to ensure at least one delivery of each task result. However, network issues can increase latency. To work around potential problems, or if you prefer synchronous processes, you can use the following APIs in either the Media pipelines SDK or the Voice SDK:

- GetSpeakerSearchTask
- GetVoiceToneAnalysisTask

> **Important**
>
> The `GetVoiceToneAnalysisTask` API only returns a task's status. It doesn't return task results. To see results, use an Amazon SQS, Amazon SNS, or AWS Lambda notification target.

The `GetSpeakerSearchTask` API gets the latest results synchronously for a task ID, delayed messages, or messages that arrive out of order. However, we recommend using notification targets and asynchronous processing. Doing so consumes fewer computing resources.

Understanding notifications

Voice analytics automatically sends events to a target when speaker search or voice tone analysis tasks start, while they run, and when they finish. You use notification targets to receive those events. We recommend using multiple notification targets if your workflow or application needs high availability.

Also, you must use an IAM role with the policies needed to access your notification targets. For more information, see Using the call analytics resource access role.

> **Note**
>
> For Amazon SQS and Amazon SNS, we do not support first-in-first-out queues. As a result, messages may arrive out of order. We recommend checking the timestamps to order messages as needed, and persisting messages in a data store such as Amazon DynamoDB. You can also use the Get APIs described in Polling for task results to receive the latest results.

The following table lists the events and their corresponding detail types.

Notification event	Detail type
Voice analytics metadata	`VoiceAnalyticsStatus`
Speaker search	`SpeakerSearchStatus`
Voice tone analysis	`VoiceToneAnalysisStatus`

Understanding IAM policies for notification targets

You must use policies in the IAM role in a Call Analytics configuration that allow access to your Amazon SQS, Amazon SNS, AWS Lambda, or Amazon KDS notification targets. For more information, see Using the call analytics resource access role in this guide.

Speaker search events

Speaker search events have the `SpeakerSearchStatus` detail type.

Amazon Chime SDK Voice Connectors send the following speaker search events:

- Identification matches
- Voice embedding generation

The events can have the following statuses:

- `IdentificationSuccessful` – Successfully identified at least one matching voice profile ID with a high confidence score in the given voice profile domain.

- `IdentificationFailure` – Failed to perform identification. Causes: the caller doesn't talk for at least 10 seconds, poor audio quality.
- `IdentificationNoMatchesFound` – Unable to find a high confidence match in the given voice profile domain. The caller may be new, or their voice may have changed.
- `VoiceprintGenerationSuccessful` – The system generated a voice embedding using 20 seconds of non-silent audio.
- `VoiceprintGenerationFailure` – The system failed to generate a voice embedding. Causes: caller doesn't talk for at least 20 seconds, poor audio quality.

Identification matches

After the StartSpeakerSearchTask API is called for a given `transactionId`, the Voice Connector service returns an identification match notification after 10 seconds of non-silent speech. The service returns the top 10 matches, along with a voice profile ID and confidence score ranging from [0, 1]. The higher the confidence score, the more likely the speaker from the call matches the voice profile ID. If the machine learning model finds no matches, the notification's `detailStatus` field contains `IdentificationNoMatchesFound`.

The following example shows notification for a successful match.

```
{
    "version": "0",
    "id": "12345678-1234-1234-1234-111122223333",
    "detail-type": "SpeakerSearchStatus",
    "service-type": "VoiceAnalytics",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "taskId": "uuid",
        "detailStatus": "IdentificationSuccessful",
        "speakerSearchDetails" : {
            "results": [
                {
                    "voiceProfileId": "vp-505e0992-82da-49eb-9d4a-4b34772b96b6",
                    "confidenceScore": "0.94567856",
                },
                {
```

```
                    "voiceProfileId": "vp-fba9cbfa-4b8d-4f10-9e41-9dfdd66545ab",
                    "confidenceScore": "0.82783350",
                },
                {
                    "voiceProfileId": "vp-746995fd-16dc-45b9-8965-89569d1cf787",
                    "confidenceScore": "0.77136436",
                }
            ]
        },
        "mediaInsightsPipelineId": "87654321-33ca-4dc6-9cdf-abcde6612345",
        "sourceArn": "arn:aws:chime:us-east-1:111122223333:media-
pipeline/87654321-33ca-4dc6-9cdf-abcde6612345",
        "streamArn": "arn:aws:kinesisvideo:us-east-1:111122223333:stream/my-
stream/0123456789012",
        "channelId": 0
    }
}
```

Voice embedding generation

After an additional 10 seconds of non-silent speech, the Voice Connector sends a voice embedding generation notification to the notification targets. You can enroll new voice embeddings in a voice profile, or update a print already in a voice profile.

The following example shows the notification for a successful match, meaning you can update the associated voice profile.

```
{
    "version": "0",
    "id": "12345678-1234-1234-1234-111122223333",
    "detail-type": "SpeakerSearchStatus",
    "service-type": "VoiceAnalytics",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "taskId": "guid",
        "detailStatus": "VoiceprintGenerationSuccess",
        "mediaInsightsPipelineId": "87654321-33ca-4dc6-9cdf-abcde6612345",
        "sourceArn": "arn:aws:chime:us-east-1:111122223333:media-
pipeline/87654321-33ca-4dc6-9cdf-abcde6612345",
```

```
        "streamArn": "arn:aws:kinesisvideo:us-east-1:111122223333:stream/my-
stream/0123456789012",
        "channelId": 0
    }
}
```

Voice tone analysis events

Voice tone analysis events have the `VoiceToneAnalysisStatus` detail type. The analyses can return these statuses:

- `VoiceToneAnalysisSuccessful` – Successfully analyzed the caller and agent voices into probabilities of sentiment—positive, negative, or neutral.
- `VoiceToneAnalysisFailure` – Failed to perform tone analysis. This can happen if the caller hangs without talking for 10 seconds, or if the audio quality becomes too poor.
- `VoiceToneAnalysisCompleted` – Successfully analyzed the user and agent voices into probabilities of sentiment for the entire call. This is the final event, sent when the voice tone analysis finishes.

The following example shows a typical voice tone analysis event.

```
{
  "detail-type": "VoiceToneAnalysisStatus",
  "service-type": "VoiceAnalytics",
  "source": "aws.chime",
  "account": "216539279014",
  "time": "2022-08-26T17:55:15.563441Z",
  "region": "us-east-1",
  "detail": {
    "taskId": "uuid",
    "detailStatus": "VoiceToneAnalysisSuccessful",
    "voiceToneAnalysisDetails": {
      "currentAverageVoiceTone": {
          "startTime": "2022-08-26T17:55:15.563Z",
          "endTime": "2022-08-26T17:55:45.720Z",
          "voiceToneLabel": "neutral",
          "voiceToneScore": {
            "neutral": "0.83",
            "positive": "0.13",
            "negative": "0.04"
          }
```

```
      },
      "overallAverageVoiceTone": {
          "startTime": "2022-08-26T16:23:13.344Z",
          "endTime": "2022-08-26T17:55:45.720Z",
          "voiceToneLabel": "positive",
          "voiceToneScore": {
            "neutral": "0.25",
            "positive": "0.65",
            "negative": "0.1"
          }
      }
    },
        "startFragmentNumber": "01234567890123456789",
        "mediaInsightsPipelineId": "87654321-33ca-4dc6-9cdf-abcde6612345",
        "sourceArn": "arn:aws:chime:us-east-1:111122223333:media-
pipeline/87654321-33ca-4dc6-9cdf-abcde6612345",
        "streamArn": "arn:aws:kinesisvideo:us-east-1:111122223333:stream/my-
stream/0123456789012",
        "channelId": 0
  },
  "version": "0",
  "id": "Id-f928dfe3-f44b-4965-8a17-612f9fb92d59"
}
```

Post-call summary events

Post call summary events are sent 5 minutes after the call has ended. These summaries provide an overview of the speaker search tasks that occurred throughout the call.

The following example shows a post call summary with the best voice profile match, confirmed speaker identity, and a list of the voice profiles created or updated through the `CreateVoiceProfile` and `UpdateVoiceProfile` API calls made during the call.

```
{
    "version": "0",
    "id": "12345678-1234-1234-1234-111122223333",
    "detail-type": "VoiceAnalyticsStatus",
    "service-type": "VoiceAnalytics",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-east-1",
    "resources": [],
```

```
    "detail": {
        "detailStatus": "PostCallVoiceAnalytics",
        "callId": "22e8dee8-bbd7-4f94-927b-2d0ebaeddc1c",
        "transactionId": "daaeb6bf-2fe2-4e51-984e-d0fbf2f09436",
        "voiceConnectorId": "abcdef1ghij2klmno3pqr4",
        "isCaller": true | false,
        "speakerSearchResults": {
            "bestMatchedVoiceProfileId": "vp-04c25ba1-a059-4fd3-8495-4ac91b55e2bf",
            "customerValidatedCallerIdentity": "vp-04c25ba1-
a059-4fd3-8495-4ac91b55e2bf",
            "createVoiceProfileTransactions": [
                {
                    "voiceProfileId": "vp-04c25ba1-a059-4fd3-8495-4ac91b55e2bf",
                    "requestTimestamp": "2022-12-14T18:38:38.796Z"
                },
                {
                    "voiceProfileId": "vp-04c25ba1-a059-4fd3-8495-4ac91b55e2bf",
                    "requestTimestamp": "2022-12-14T18:38:38.796Z",
                }
            ],
            "updateVoiceProfileTransactions": [
                {
                    "voiceProfileId": "vp-04c25ba1-a059-4fd3-8495-4ac91b55e2bf",
                    "requestTimestamp": "2022-12-14T18:38:38.796Z",
                },
                {
                    "voiceProfileId": "vp-04c25ba1-a059-4fd3-8495-4ac91b55e2bf",
                    "requestTimestamp": "2022-12-14T18:38:38.796Z",
                }
            ]
        }
    }
}
```

Voice analytics example Lambda function

The Python code in the following example processes notifications received from a Voice Connector. You can add the code to an AWS Lambda function. You can also use it to trigger your Amazon SQS queue, Amazon SNS topic, or Amazon Kinesis Data Stream. You can then store the notifications in an `EventTable` for future processing. For the exact notification formats, see Understanding notifications.

```
import base64
```

```
import boto3
import json
import logging
import time

from datetime import datetime
from enum import Enum

log = logging.getLogger()
log.setLevel(logging.INFO)

dynamo = boto3.client("dynamodb")

EVENT_TABLE_NAME = "EventTable"

class EventType(Enum):
    """
    This example code uses a single Lambda processor to handle either
    triggers from SQS, SNS, Lambda, or Kinesis. You can adapt it to fit your
    desired infrastructure depending on what you prefer. To distinguish
    where we get events from, we use an EventType enum as an
    example to show the different ways of parsing the notifications.
    """
    SQS = "SQS"
    SNS = "SNS"
    LAMBDA = "LAMBDA"
    KINESIS = "KINESIS"


class AnalyticsType(Enum):
    """
    Define the various analytics event types that this Lambda will
    handle.
    """
    SPEAKER_SEARCH = "SpeakerSearch"
    VOICE_TONE_ANALYSIS = "VoiceToneAnalysis"
    ANALYTICS_READY = "AnalyticsReady"
    UNKNOWN = "UNKNOWN"


class DetailType(Enum):
    """
    Define the  various detail types that Voice Connector's voice
    analytics feature can return.
```

```
    """
    SPEAKER_SEARCH_TYPE = "SpeakerSearchStatus"
    VOICE_TONE_ANALYSIS_TYPE = "VoiceToneAnalysisStatus"
    ANALYTICS_READY = "VoiceAnalyticsStatus"


def handle(event, context):
    """
    Example of how to handle incoming Voice Analytics notification messages
    from Voice Connector.
    """
    logging.info(f"Received event of type {type(event)} with payload {event}")
    is_lambda = True

    # Handle triggers from SQS, SNS, and KDS. Use the below code if you would like
    # to use this Lambda as a trigger for an existing SQS queue, SNS topic or Kinesis
    # stream.
    if "Records" in event:
        logging.info("Handling event from SQS or SNS since Records exists")
        is_lambda = False
        for record in event.get("Records", []):
            _process_record(record)

    # If you would prefer to have your Lambda invoked directly, use the
    # below code to have the Voice Connector directly invoke your Lambda.
    # In this scenario, there are no "Records" passed.
    if is_lambda:
        logging.info(f"Handling event from Lambda")
        event_type = EventType.LAMBDA
        _process_notification_event(event_type, event)


def _process_record(record):
    # SQS and Kinesis use eventSource.
    event_source = record.get("eventSource")

    # SNS uses EventSource.
    if not event_source:
        event_source = record.get("EventSource")

    # Assign the event type explicitly based on the event source value.
    event_type = None
    if event_source == "aws:sqs":
        event = record["body"]
```

```
        event_type = EventType.SQS
    elif event_source == "aws:sns":
        event = record["Sns"]["Message"]
        event_type = EventType.SNS
    elif event_source == "aws:kinesis":
        raw_data = record["kinesis"]["data"]
        raw_message = base64.b64decode(raw_data).decode('utf-8')
        event = json.loads(raw_message)
        event_type = EventType.KINESIS
    else:
        raise Exception(f"Event source {event_source} is not supported")

    _process_notification_event(event_type, event)


def _process_notification_event(
    event_type: EventType,
    event: dict
):
    """
    Extract the attributes from the Voice Analytics notification message
    and store it as a DynamoDB item to process later.
    """
    message_id = event.get("id")
    analytics_type = _get_analytics_type(event.get("detail-type"))
    pk = None
    if analytics_type == AnalyticsType.ANALYTICS_READY.value or analytics_type ==
 AnalyticsType.UNKNOWN.value:
        transaction_id = event.get("detail").get("transactionId")
        pk =
 f"transactionId#{transaction_id}#notificationType#{event_type.value}#analyticsType#{analytics_
    else:
        task_id = event.get("detail").get("taskId")
        pk =
 f"taskId#{task_id}#notificationType#{event_type.value}#analyticsType#{analytics_type}"
    logging.info(f"Generated PK {pk}")
    _create_request_record(pk, message_id, json.dumps(event))


def _create_request_record(pk: str, sk: str, body: str):
    """
    Record this notification message into the Dynamo db table
    """
    try:
```

```
        # Use consistent ISO8601 date format.
        # 2019-08-01T23:09:35.369156 -> 2019-08-01T23:09:35.369Z
        time_now = (
            datetime.utcnow().isoformat()[:-3] + "Z"
        )
        response = dynamo.put_item(
            Item={
                "PK": {"S": pk},
                "SK": {"S": sk},
                "body": {"S": body},
                "createdOn": {"S": time_now},
            },
            TableName=EVENT_TABLE_NAME,
        )
        logging.info(f"Added record in table {EVENT_TABLE_NAME}, response :
 {response}")
    except Exception as e:
        logging.error(f"Error in adding record: {e}")


def _get_analytics_type(detail_type: str):
    """
    Get analytics type based on message detail type value.
    """
    if detail_type == DetailType.SPEAKER_SEARCH_TYPE.value:
        return AnalyticsType.SPEAKER_SEARCH.value
    elif detail_type == DetailType.VOICE_TONE_ANALYSIS_TYPE.value:
        return AnalyticsType.VOICE_TONE_ANALYSIS.value
    elif detail_type == DetailType.ANALYTICS_READY.value:
        return AnalyticsType.ANALYTICS_READY.value
    else:
        return AnalyticsType.UNKNOWN.value
```

Important

You must receive consent before you call the StartSpeakerSearchTask or StartVoiceToneAnalysis APIs. We recommend that you persist the events in a holding area, such as Amazon DynamoDB, until you receive consent.

Understanding data storage, opt-out, and data-retention policies

The Amazon Chime SDK uses voice data to provide and improve the speaker search service. As part of that, we use enrollment audio, the recorded snippets used to create voice embeddings, to train our machine learning and artificial intelligence models. You can opt out of having your data used to train the models, and the topics in this section explain how.

Topics

Understanding data storage for speaker search

The Amazon Chime SDK stores the following data for speaker search:

- The voice embeddings attached to the voice profiles that we use to provide the speaker search functionality.
- Enrollment audio, the recorded snippets of speech used to create the voice embeddings for each voice profile. We use the enrollment audio recordings to:
  - Keep the speaker search models up to date, a critical part of providing the speaker search feature.
  - Train the machine learning model to develop and improve the service. The use of enrollment audio for training is optional, and you can opt out of this use by selecting an opt-out policy as described in the following section.

Handling opt outs for speaker search

You can handle opt outs for end users and entire organizations. Opting out has the following effects:

- After you opt out, voice analytics will not use any new enrollment audio for model training, and it will not use any enrollment audio collected and stored prior to your opting out.
- After you opt out, voice analytics will store and use enrollment audio in order to provide the speaker search service.

> **Warning**
>
> The following opt-out actions are irreversible. You can't recover deleted data.

Handling end user opt-outs

When end users want to opt out of speaker search, call the DeleteVoiceProfile API. This action removes the voice profile, plus the voice embeddings and enrollment audio.

To delete a group of voice embeddings, call the DeleteVoiceProfileDomain API to remove the domain. This action deletes *all* the voice profiles in a domain.

Handling opt-out at the organizational level

To handle opt outs for an entire organization, use an AWS Organizations opt-out policy. Use the `chimesdkvoiceanalytics` service name. For information about the policies, see AI services opt-out policies in the *AWS Organizations User Guide*.

> **Note**
>
> To use an opt-out policy, your AWS accounts must be centrally managed by AWS Organizations. If you haven't already created an organization for your AWS accounts, see Creating and managing an organization in the *AWS Organizations User Guide*.

Understanding data retention for Amazon Chime SDK voice analytics

By default, Amazon Chime SDK voice analytics deletes voice embeddings after 3 years. We do this because people’s voices change over time, and also for security. You can use the UpdateVoiceProfile API to update expired voice embeddings.

Results from StartSpeakerSearchTask and StartVoiceToneAnalysisTask will also be available from their respective GetSpeakerSearchTask and GetVoiceToneAnalytisTask APIs for a maximum of 7 days.

Voice embeddings generated from a StartSpeakerSearchTask are available for persistence via the CreateVoiceProfile and UpdateVoiceProfile APIs for 24 hours, after which they're deleted and not available.

To remove results, and to handle consent withdrawals from your customers, see the previous section.

Using voice APIs to run voice analytics

For backwards compatibility, you can use Amazon Chime SDK Voice APIs to start and manage voice analytics. However, only the media insights pipeline APIs for voice analytics provide new features, so we strongly recommend using them instead.

The following sections explain the differences between the voice and media insights pipelines APIs.

Stopping tasks

If you use a Voice Connector to start voice analytics tasks, and you then use the UpdateMediaInsightsPipelineStatus API to pause the pipeline, the tasks continue running. To stop the tasks, you must call the StopSpeakerSearchTask and StopVoiceToneAnalysisTask APIs.

Understanding the notification differences

When you use voice APIs to run voice analytics, the notifications differ from those generated by media insights pipelines.

- Voice analytics ready events are only available for tasks started using voice APIs.
- You need to use the `voiceConnectorId`, `transactionId`, or `callId` fields in your notifications to associate a voice analytics task with a call. If you use media insights pipelines to run voice analytics, you use the `mediaInsightsPipelineId` and `streamArn` or `channelId` fields to associate a task with a call.

The following topics explain how to use notifications with voice APIs.

Topics

- Voice analytics ready events
- Speaker search events
- Voice tone analysis events

Voice analytics ready events

Voice analytics ready events have the `VoiceAnalyticsStatus` detail type.

You use Amazon Chime SDK Voice Connectors to start analytics tasks. When your receive a voice analytics ready event, you can trigger a speaker search or voice tone analysis task for the call, identified by the following properties:

- `voiceConnectorId`
- `transactionId`

> **Note**
>
> This notification is provided only when you have a media insights pipeline configuration with voice analytics enabled and associated with a Voice Connector. This notification is NOT provided when customers call the `CreateMediaInsightsPipeline` API and launch a speaker search task or voice tone analysis task via the Media Pipelines SDK.

The SIP headers returned by a Voice Connector contain the `transactionId`. If you don't have access to the SIP headers, the `AnalyticsReady` notification event also contains the `voiceConnectorId` and `transactionId`. That allows you to programmatically receive the information and call the StartSpeakerSearchTask, or StartVoiceToneAnalysisTask APIs.

When voice analytics is ready for processing, the Voice Connector sends an event with `"detailStatus": "AnalyticsReady"` to the notification target as a JSON body. If you use Amazon SNS or Amazon SQS, that body appears in the “Records” field in the Amazon SNS or Amazon SQS payload.

The following example shows a typical JSON body.

```
{
    "detail-type": "VoiceAnalyticsStatus",
    "version": "0",
    "id": "Id-f928dfe3-f44b-4965-8a17-612f9fb92d59",
    "source": "aws.chime",
    "account": "123456789012",
    "time": "2022-08-26T17:55:15.563441Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "detailStatus": "AnalyticsReady",
        "callDetails": {
```

```
            "isCaller": false,
            "transactionId": "daaeb6bf-2fe2-4e51-984e-d0fbf2f09436",
            "voiceConnectorId": "fuiopl1fsv9caobmqf2vy7"
        }
    }
}
```

This notification allows you to trigger additional callbacks to your application, and to handle any legal requirements, such as notice and consent, prior to calling the voice analytics task APIs.

Speaker search events

Speaker search events have the `SpeakerSearchStatus` detail type.

Amazon Chime SDK Voice Connectors send the following speaker search events:

- Identification matches
- Voice embedding generation

The events can have the following statuses:

- `IdentificationSuccessful` – Successfully identified at least one matching voice profile ID with a high confidence score in the given voice profile domain.
- `IdentificationFailure` – Failed to perform identification. Causes: the caller doesn't talk for at least 10 seconds, poor audio quality.
- `IdentificationNoMatchesFound` – Unable to find a high confidence match in the given voice profile domain. The caller may be new, or their voice may have changed.
- `VoiceprintGenerationSuccessful` – The system generated a voice embedding using 20 seconds of non-silent audio.
- `VoiceprintGenerationFailure` – The system failed to generate a voice embedding. Causes: caller doesn't talk for at least 20 seconds, poor audio quality.

Identification matches

After the StartSpeakerSearchTask API is called for a given `transactionId`, the Voice Connector service returns an identification match notification after 10 seconds of non-silent speech. The service returns the top 10 matches, along with a voice profile ID and confidence score ranging from [0, 1]. The higher the confidence score, the more likely the speaker from the call matches the voice

profile ID. If the machine learning model finds no matches, the notification's `detailStatus` field contains `IdentificationNoMatchesFound`.

The following example shows notification for a successful match.

```
{
    "version": "0",
    "id": "12345678-1234-1234-1234-111122223333",
    "detail-type": "SpeakerSearchStatus",
    "service-type": "VoiceAnalytics",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "taskId": "uuid",
        "detailStatus": "IdentificationSuccessful",
        "speakerSearchDetails" : {
            "results": [
                {
                    "voiceProfileId": "vp-505e0992-82da-49eb-9d4a-4b34772b96b6",
                    "confidenceScore": "0.94567856",
                },
                {
                    "voiceProfileId": "vp-fba9cbfa-4b8d-4f10-9e41-9dfdd66545ab",
                    "confidenceScore": "0.82783350",
                },
                {
                    "voiceProfileId": "vp-746995fd-16dc-45b9-8965-89569d1cf787",
                    "confidenceScore": "0.77136436",
                }
            ]
        },
        "isCaller": false,
        "voiceConnectorId": "abcdef1ghij2klmno3pqr4",
        "transactionId": "daaeb6bf-2fe2-4e51-984e-d0fbf2f09436"
    }
}
```

Voice embedding generation

After an additional 10 seconds of non-silent speech, the Voice Connector sends a voice embedding generation notification to the notification targets. You can enroll new voice embeddings in a voice profile, or update a print already in a voice profile.

The following example shows the notification for a successful match, meaning you can update the associated voice profile.

```
{
    "version": "0",
    "id": "12345678-1234-1234-1234-111122223333",
    "detail-type": "SpeakerSearchStatus",
    "service-type": "VoiceAnalytics",
    "source": "aws.chime",
    "account": "111122223333",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "taskId": "guid",
        "detailStatus": "VoiceprintGenerationSuccess",
        "isCaller": false,
        "transactionId": "12345678-1234-1234",
        "voiceConnectorId": "abcdef1ghij2klmno3pqr"
    }
}
```

Voice tone analysis events

Voice tone analysis events have the `VoiceToneAnalysisStatus` detail type. The analyses can return these statuses:

- `VoiceToneAnalysisSuccessful` – Successfully analyzed the caller and agent voices into probabilities of sentiment—positive, negative, or neutral.
- `VoiceToneAnalysisFailure` – Failed to perform tone analysis. This can happen if the caller hangs without talking for 10 seconds, or if the audio quality becomes too poor.
- `VoiceToneAnalysisCompleted` – Successfully analyzed the user and agent voices into probabilities of sentiment for the entire call. This is the final event, sent when the voice tone analysis finishes.

The following example shows a typical voice tone analysis event.

```
{
  "detail-type": "VoiceToneAnalysisStatus",
  "service-type": "VoiceAnalytics",
  "source": "aws.chime",
  "account": "216539279014",
  "time": "2022-08-26T17:55:15.563441Z",
  "region": "us-east-1",
  "detail": {
    "taskId": "uuid",
    "detailStatus": "VoiceToneAnalysisSuccessful",
    "voiceToneAnalysisDetails": {
      "currentAverageVoiceTone": {
          "startTime": "2022-08-26T17:55:15.563Z",
          "endTime": "2022-08-26T17:55:45.720Z",
          "voiceToneLabel": "neutral",
          "voiceToneScore": {
            "neutral": "0.83",
            "positive": "0.13",
            "negative": "0.04"
          }
      },
      "overallAverageVoiceTone": {
          "startTime": "2022-08-26T16:23:13.344Z",
          "endTime": "2022-08-26T17:55:45.720Z",
          "voiceToneLabel": "positive",
          "voiceToneScore": {
            "neutral": "0.25",
            "positive": "0.65",
            "negative": "0.1"
          }
      }
    },
    "isCaller": true,
    "transactionId": "daaeb6bf-2fe2-4e51-984e-d0fbf2f09436",
    "voiceConnectorId": "fuiopl1fsv9caobmqf2vy7"
  },
  "version": "0",
  "id": "Id-f928dfe3-f44b-4965-8a17-612f9fb92d59"
}
```

Call analytics service quotas

The tables in this section list the service quotas for Amazon Chime SDK call analytics.

For more information about the call analytics Regions, refer to Available AWS Regions for the Amazon Chime SDK service, earlier in this guide.

Amazon Chime SDK call analytics and voice analytics have the following service quotas.

Resource	Default limit	Adjustable
Media Insights Pipeline Configurations per region	100	Yes
Active Media Insights Pipelines per region	20	Yes
Voice profile domains per region	3	Yes
Voice profiles per voice profile domain	20	Yes
Active speaker search tasks per region	25	Yes
Active voice tone analysis tasks per region	25	Yes
Active Voice Connector calls with voice analytics per region	25	Yes
Active speaker search tasks per Voice Connector call per transaction ID	1	No
Active voice tone analysis task per Voice Connector call per transaction ID	1	No

Resource	Default limit	Adjustable
Maximum concurrent API calls per voice profile domain	1	Yes
Maximum concurrent API calls per voice profile	1	Yes
Maximum concurrent API calls per speaker search task	1	Yes
Maximum concurrent API calls per voice tone analysis task	1	Yes

For more information about API rates and quotas, refer to Amazon Chime SDK endpoints and quotas in the *AWS General Reference*.

> **Note**
>
> If you exceed the quota for any Region, you receive a **Resource Limit Exceeded** exception. You can use the **Service Quotas** page in the AWS console to request an increase, or you can contact your customer support representative.
> Several of the call analytics APIs create resources and API requests for other AWS services. Those additional count against your account's quotas. If you request a quota or transactions-per-second increase from call analytics, you must also request increases for those other AWS services. Otherwise, your requests may be throttled and fail.

Using the Amazon Chime SDK client library for Android

Currently, you'll find the Amazon Chime SDK client library for Android on GitHub. Go to https://github.com/aws/amazon-chime-sdk-android.

Using the Amazon Chime SDK client library for iOS

Currently, you'll find the Amazon Chime SDK client library for iOS on GitHub. Go to https://github.com/aws/amazon-chime-sdk-ios.

Using the Amazon Chime SDK client library for JavaScript

This guide provides a conceptual overview of the Amazon Chime SDK client library for JavaScript, and example code for critical server and client components.

Topics

Understanding the components of an Amazon Chime SDK application

To embed real-time audio, video, and screen-sharing capabilities into your Amazon Chime SDK applications, you use these components:

- **The Amazon Chime SDK client library for JavaScript**, the client-side SDK that you integrate into your browser or Electron web application. You do that by adding the Amazon Chime SDK for JavaScript NPM package as a dependency. This package leverages the MediaDevices and WebRTC APIs to join meetings and exchange audio, video, and to share content with other attendees. It gives you a control surface for managing the different types of media and the ability to bind those resources to your application's user interfaces.
- **The AWS SDK**, the Amazon Chime SDK API that your server application uses to authenticate and authorize meeting requests from your web application. The AWS SDK gives you API actions such

as chime:CreateMeeting and chime:CreateAttendee to create and manage meeting and attendee resources.

Like any other AWS resource, the AWS Identity and Access Management (IAM) service configures access to these actions. The AWS SDK is available in several programming languages and takes the complexity out of calling the AWS SDK Chime API from your server application. If your application does not currently use a server application, you can start with the AWS CloudFormation template included in the demos/serverless folder. That demo shows you how to build an AWS Lambda-based serverless application that uses the AWS SDK Chime API.

- **Amazon Chime SDK media services** provides the audio, video, and signaling that the Amazon Chime SDK client library for JavaScript uses to connect to meetings. Media services are available globally to support audio mixing, video forwarding, and NAT traversal using TURN relays. The Amazon Chime service team deploys, monitors, and manages these services. The media services are hosted in a single range of IP addresses – 99.77.128.0/18 – and use ports TCP/443 and UDP/3478 to simplify firewall configurations for IT administrators. Finally, these services leverage the AWS Global Cloud Infrastructure.

Understanding key concepts of the Amazon Chime SDK client library for JavaScript

To fully understand how to create and manage meetings and users, you need to understand these concepts:

Meeting – A multi-party media session. Every meeting has a unique meeting identifier. You can create meetings in one of the supported AWS Regions. When you create a meeting, a list of Media URLs are returned. Those are a key part of the data needed to join the meeting, and you need to disseminate that data to all users trying to join the meeting.

Attendee – A user trying to join a multi-party media session. Every attendee has a unique identifier, an external user identifier that can be passed in to map the attendee to a user in the developer's system, plus a signed join token that grants them access to the meeting.

MeetingSession and (DefaultMeetingSession) – The root object of the Amazon Chime SDK client library for JavaScript that represents each user’s session in a meeting. The web applications start by instantiating MeetingSession and configuring it with the right meeting and attendee information.

MeetingSessionConfiguration – Stores the meeting and attendee data needed to join a meeting session. That data is the response of the `CreateMeeting` and `CreateAttendee` API calls made by the server application. The server application passes this data to the web application, which uses it to instantiate the `MeetingSession`.

DeviceController (DefaultDeviceController) – Used to enumerate the list of available audio and video devices on a user's system. You can also use the device controller during a meeting to switch active devices.

AudioVideoFacade (DefaultAudioVideoFacade) – The key interface that powers a meeting. It provides the APIs that start, control, and end a meeting. It also provides APIs that listen for the key events that drive user experience changes, such as a roster of attendees, by tracking users joining or leaving, being muted or unmuted, actively speaking, or having poor connectivity. You can also use those APIs to bind the audio control HTML element to the meeting's audio output and play it through the selected audio output device.

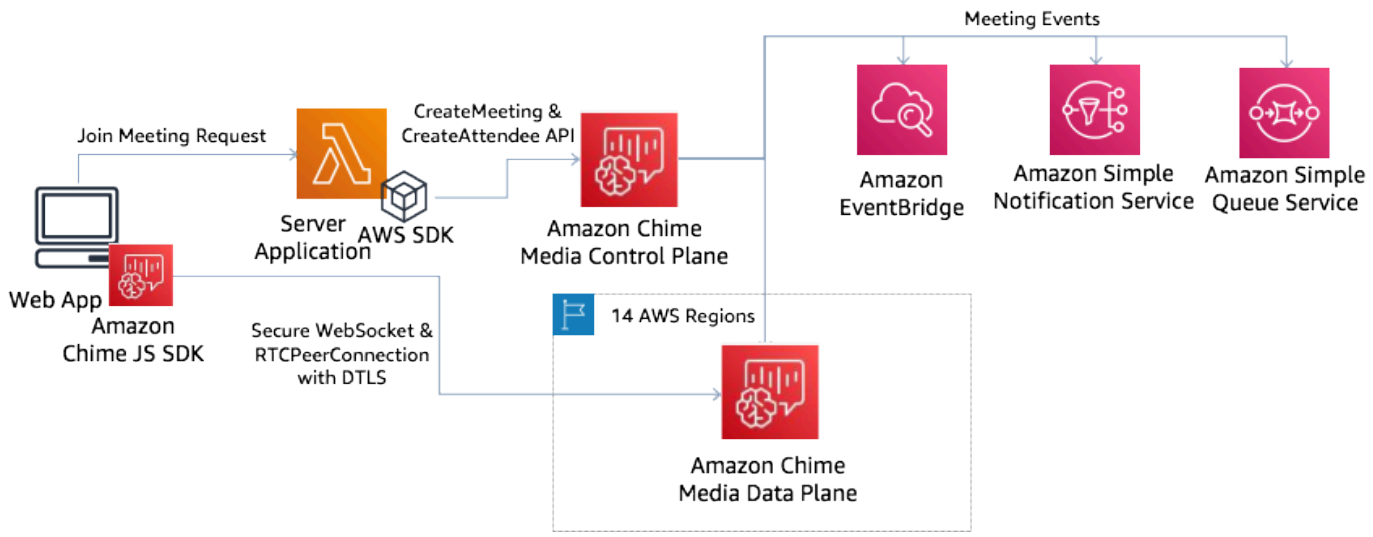
ActiveSpeakerDetectorFacade (DefaultActiveSpeakerDetector) – The API that subscribes to active speaker events. Periodically returns a list of attendees ordered by their mic volume over time. You can override and tweak the active speaker policy as needed.

ContentShareController (DefaultContentShareController) – APIs that start-stop and pause-unpause content sharing. It also provides APIs to listen to lifecycle events to track content sharing status.

Logger (ConsoleLogger) – The interface used to leverage the console logs, or pass in a logger object to override the current logging implementation and get different levels of logs from the Amazon Chime SDK.

Understanding service architecture

This high-level architecture diagram shows how the components listed in Understanding key concepts of the Amazon Chime SDK client library for JavaScriptinteract and work with other AWS services:



Understanding web application architecture

You can serve your web application from a content delivery network, and load it when the user navigates to a URL in a browser. You can also wrap it in a platform-native Electron application that the user installs on their machine.

To join a new or existing meeting, the web application makes REST requests to the server application. Typically, the requests carry an authorization token or a cookie that your application uses for other API requests. You can also design your web client to send a region hint to the server, which the latter can use when providing the MediaRegion parameter to chime:CreateMeeting. Your web application can determine the closest media services region by making an HTTP GET request to the https://nearest-media-region.l.chime.aws endpoint.

Understanding server application architecture

When a server receives a request from a client, it first ensures that the user is authorized to start or join a meeting. The server uses the embedded AWS SDK in the language of choice to make chime:CreateMeeting and chime:CreateAttendee API calls to the global media control plane. It does that to create the meeting and attendees in one of the supported AWS Regions. To make these requests, the service needs the appropriate IAM user or role. In turn, IAM users and roles need the AmazonChimeSDK policy.

Understanding the Amazon Chime SDK media control plane

The Amazon Chime SDK media control plane is global – hosted out of us-east-1 – and serves the chime:CreateMeeting and chime:CreateAttendee APIs used to create and manage meeting and attendee resources across the data plane. It validates credentials and ensures the session is bootstrapped in the data plane in the requested region.

The control plane also triggers Amazon Chime SDK Events to the notification mechanisms such as Amazon EventBridge, Amazon Simple Queueing Service (SQS) or Amazon Simple Notification Service (SNS). AWS constantly monitors the services, and they scale automatically as load increases. The APIs are designed to only accept opaque user identifiers and not user data, so they adhere to data-sovereignty requirements.

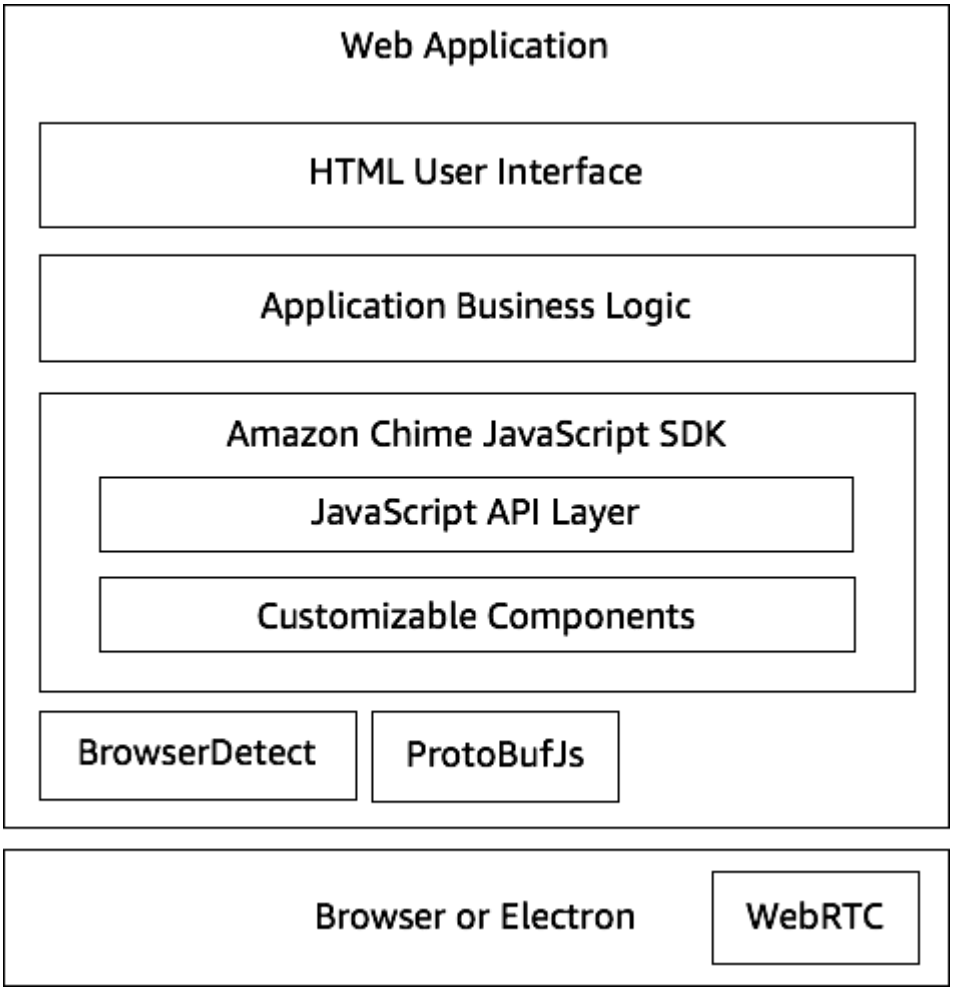
Understanding the Amazon Chime SDK media data plane

You can use any control plane region to create meetings in all AWS Regions. The media data plane is available in all AWS Regions. It includes an audio mixing service, video forwarding service, TURN service and Session Initiation Protocol (SIP) interoperability services. The services are constantly monitored and are designed to scale automatically as load increases. To learn more, see Amazon Chime SDK media Regions .

For a current list of Regions and Availability Zones, see Regions and Availability Zones .

Understanding web application component architecture

This diagram shows the architecture of an Amazon Chime SDK web client application:



A web application typically consists of an HTML and CSS user interface layer powered by the application business logic layer. You can build the web application in plain HTML and JavaScript, or you can use UI frameworks such as React and Angular.

The web application's business logic layer interacts with the Amazon Chime SDK client library for JavaScript through a set of JavaScript APIs. The DefaultMeetingSession is the root object of the SDK. When building a server application you use MeetingSessionConfiguration to initialize it with meeting and attendee information and join the meeting. The DefaultMeetingSession also exposes the AudioVideoFacade, which enables the business logic layer to take actions, and to register callbacks that update the user interface when the underlying state of the session changes.

The Amazon Chime SDK client library for JavaScript is open-source and has a set of customizable components that you can override as needed. The default implementations allow you to build a complete unified communications application such as our demo MeetingV2 application. The Amazon Chime SDK client library for JavaScript depends on two other libraries:

- Browser-Detect for identifying the browser type and capabilities.
- ProtoBufJs to encode and decode signaling commands and responses needed to join a media sessions.

The Amazon Chime SDK also depends on the browser or Electron application to provide the Device Management APIs and the WebRTC implementation for an audio-video session.

The source Amazon Chime SDK client library for JavaScript is in TypeScript, but you can use the TypeScript compiler to compile it to JavaScript. You can then bundle it using a module bundler such as Webpack. As a best practice, install the Amazon Chime SDK client library for JavaScript from the NPM registry, and then use it in a CommonJS environment. AWS also provides a rollup script for bundling the Amazon Chime SDK into a minified JS file in case you want to directly include it as a script tag in your HTML.

Building a server application

The information in the following section explains how to build an Amazon Chime SDK server application. Each section provides example code as needed, and you can adapt that code to meet your needs.

Topics

Creating IAM users or roles

You create users as IAM users, or in roles appropriate to your use case. You then assign the following policy to them. This ensures that you have the necessary permissions for the AWS SDK embedded in your server application. In turn, that allows you to perform lifecycle operations on the meeting and attendee resources.

```
  // Policy ARN:     arn:aws:iam::aws:policy/AmazonChimeSDK
   // Description:    Provides access to Amazon Chime SDK operations
   {
   "Version": "2012-10-17",
   "Statement": [
       {
           "Action": [
               "chime:CreateMeeting",
               "chime:DeleteMeeting",
               "chime:GetMeeting",
               "chime:ListMeetings",
               "chime:CreateAttendee",
               "chime:BatchCreateAttendee",
               "chime:DeleteAttendee",
               "chime:GetAttendee",
               "chime:ListAttendees"
           ],
           "Effect": "Allow",
           "Resource": "*"
    }
]}
```

Configuring the AWS SDK to invoke the APIs

This code sample shows you how to pass credentials to the AWS SDK, and set a region and endpoint.

```
AWS.config.credentials = new AWS.Credentials(accessKeyId, secretAccessKey, null);
const chime = new AWS.Chime({ region: 'us-east-1' });
chime.endpoint = new AWS.Endpoint('https://service.chime.aws.amazon.com/console');
```

Creating a meeting

A CreateMeeting API call accepts a required parameter, the `ClientRequestToken`, that allows developers to pass in a uniqueness context. It also accepts optional parameters such as `MediaRegion`, which represents the media services data plane region to choose for the meeting, the `MeetingHostId` used to pass in an opaque identifier to represent the meeting host, if applicable, and the `NotificationsConfiguration` for receiving meeting lifecycle events.

By default, Amazon EventBridge delivers the events. Optionally, you can also receive events by passing an SQS queue ARN or an SNS Topic ARN in `NotificationsConfiguration`. The API Returns a Meeting object that contains a unique `MeetingId`, plus the `MediaRegion` and the `MediaPlacement` object with a set of media URLs.

```
meeting = await chime.createMeeting({
             ClientRequestToken: clientRequestToken,
             MediaRegion: mediaRegion,
             MeetingHostId: meetingHostId,
             NotificationsConfiguration: {
                SqsQueueArn: sqsQueueArn,
                SnsTopicArn: snsTopicArn
             }
          }).promise();
```

Creating an attendee

After you create a meeting, you create an attendee resource that represents each user trying to join the media session. The CreateAttendee API takes the following:

- The `MeetingId` of the meeting to which you're adding the user.
- An `ExternalUserId`, which can be any opaque user identifier from your identity system.

For example, if you use Active Directory (AD), this can be the object ID of the user in the AD. The `ExternalUserId` is valuable because it’s passed back to the client applications when they receive attendee events from the client SDKs. This allows the client application to know who joined or left the meeting and retrieve additional information from the server application about that user, such as a display name, email, or a picture.

Calls to the `CreateAttendee` API result in an `Attendee` object. The object contains a unique `AttendeeId` that is generated by the service, the `ExternalUserId` that was passed in, and a signed `JoinToken` that allows the attendee to access the meeting for its duration, or until the DeleteAttendee API deletes the attendee.

```
      attendee = await chime.createAttendee({
               MeetingId: meeting.MeetingId,
```

```
                ExternalUserId: externalUserId,
              }).promise();
```

Sending a response to the client

Once you create the meeting and attendee resources, the server application should encode and send the meeting and attendee objects back to the client application. The client needs those pieces of information to bootstrap the Amazon Chime SDK client library for JavaScript, and enable an attendee to join the meeting successfully from a web or Electron based application.

Building a client application

To build a client application, follow the steps listed in the Amazon Chime JavaScript SDK API Overview on GitHub. The overview provides example code as needed.

Integrating background filters into a client application

This section explains how to programmatically filter video backgrounds by using background blur 2.0 and background replacement 2.0. To add a background filter to a video stream, you create a `VideoFxProcessor` that contains a `VideoFxConfig` object. You then insert that processor into a `VideoTransformDevice`.

The background filter processor uses a TensorFlow Lite machine learning model, JavaScript Web Workers, and WebAssembly to apply a filter to the background of each frame in the video stream. These assets are downloaded at runtime when you create a `VideoFxProcessor`.

The browser demo application on GitHub uses the new background blur and replacement filters. To try them, launch the demo with `npm run start`, join the meeting, then click the camera to enable video. Open the **Apply Filter** menu

()
and choose one of the **Background Blur 2.0** or **Background Replacement 2.0** options.

Topics

About using background filters

Background filters can be CPU-intensive and GPU-intensive. Some mobile devices and lower-specification laptop or desktop computers may not have the power to run the filters along with multiple video streams.

SIMD support

Background filters are more efficient in environments that support Single Instruction, Multiple Data (SIMD). The filters use less CPU for a given complexity level when you enable SIMD. Low-powered devices running browsers without SIMD support may not run background filters.

WebGL2 support

The `VideoFxProcessor` object requires browsers that support WebGL2 in order to access the GPU on the client device.

Content delivery and bandwidth

An Amazon content delivery network loads the machine-learning-model files for background filters at runtime. This provides low-latency global distribution without the need to serve a full suite of files as part of your application. However, loading model files can add latency to parts of your application. To help mitigate that impact, browsers cache the model files indefinitely. That cache makes subsequent loads significantly faster. As a best practice, check for supported browsers, then create the background filter resources when users may not notice any latency. For example, you can download model files while users wait in a lobby, or while they use a device picker.

Your application must connect to the following:

- Amazon Chime SDK media services.
- Amazon CloudFront via HTTPS (port 443).

All requests are to subdomains of `sdkassets.chime.aws`. Applications that can't access the content delivery network or don't include the correct domain in their content security policy will fail their support checks and be unable to use the filters.

For more information about CloudFront's IP address ranges, see Locations and IP address ranges of CloudFront edge servers in the *Amazon CloudFront Developer Guide*.

Browser compatibility

The following table lists the browsers and version that support background filters.

Browser	Minimum supported version
Firefox	76+
Chromium-based browsers and environments, including Edge and Electron	78+
Android Chrome	110+
Safari on macOS	16.3+
Safari on iOS (iPhone, iPad)	16.x
Chrome on iOS	110.0.0.x.x

Browser	Minimum supported version
Firefox on iOS (iPhone iPad)	16.x

Version 3.14 of the `VideoFxProcessor` object supports Android. For Android device support on versions prior to 3.14, use the `BackgroundBlurVideoFrameProcessor` and `BackgroundReplacementVideoFrameProcessor` objects. For more information about using them, refer to the backgroundfilter_video_processor page on GitHub.

Using a content-security policy

Modern web applications use a content security policy to protect users from certain classes of attacks. Applications that use the VideoFxProcessor must include the following policy directives. The directives give the Amazon Chime SDK access to the resources it needs at runtime.

Topics

- Understanding required content security policy directives
- Using the cross-origin opener policy

Understanding required content security policy directives

You must use the following content security policy directives.

- `script-src:` add `blob: https://*.sdkassets.chime.aws` to load video processing code, and `wasm-unsafe-eval` to allow running it.
- `script-src-elem:` add `blob: https://*.sdkassets.chime.aws` to load video processing code from the source.
- `worker-src:` add `blob: https://*.sdkassets.chime.aws` to load worker JavaScript across origins.

If you omit any of these entries, or if you use HTTP headers and `http-equiv` meta tags to specify a policy and inadvertently exclude any of these by intersection, then a background filter will not be able to initialize. The filter appears to be unsupported, or it creates a no-op video frame processor. You will see errors in your browser console such as:

```
Refused to connect to
```

```
'https://static.sdkassets.chime.aws/bgblur/workers/worker.js…'
because it violates the document's content security policy.
```

Understanding the required script policy directives

To function, the `VideoFxProcessor` class must load JavaScript classes at runtime from an Amazon content delivery network. These classes use WebGL2 to implement post-processing for video. To allow an application to fetch and run these classes, you must include the following directives:

- `script-src 'self' blob: https://*.sdkassets.chime.aws`
- `script-src-elem 'self' blob: https://*.sdkassets.chime.aws`

> **Note**
>
> For full support on Safari and Firefox, you must use the `script-src` and `script-src-elem` directives.

Understanding the worker policy directive

The `VideoFxProcessor` loads JavaScript classes as a blob to run a web worker thread. The thread uses machine learning models to processes video. To grant an application access for fetching and using this worker, include the following directive:

```
worker-src 'self' blob: https://*.sdkassets.chime.aws
```

Understanding the WebAssembly policy

The `VideoFxProcessor` loads a WebAssembly (WASM) module from the same Amazon-owned content delivery network. In Chrome 95 and later, compiled WASM modules cannot be passed across multiple module boundaries. To allow fetching and instantiating these modules, include `'wasm-unsafe-eval'` in the `script-src` directive.

For more information about the Content Security Policy's documentation for WebAssembly, refer to [WebAssembly Content Security Policy]() on GitHub.

(Optional) Understanding the background image policy

To use a dynamically loaded background image with a background replacement filter, the `VideoFxProcessor` must have access to the image. To do that, include a `connect-src` directive with the domain that hosts the image.

Understanding the content security policy

The following example policy allows you to use the `VideoFxProcessor`. The `connect-src` definitions aren't specific to a `VideoFxProcessor`. Instead, they're related to the audio and video from an Amazon Chime SDK meeting.

```
<head>
    <meta http-equiv="Content-Security-Policy"
        content="base-uri 'self';
        connect-src       'self' https://*.chime.aws wss://*.chime.aws https://
*.amazonaws.com wss://*.chime.aws https://*.ingest.chime.aws;
        script-src        'self' blob: 'wasm-unsafe-eval' https://
*.sdkassets.chime.aws;
        script-src-elem   'self' blob: https://*.sdkassets.chime.aws;
        worker-src        'self' blob: https://*.sdkassets.chime.aws;">
</head>
```

Understanding content security policy errors

If you omit any of the required directives, the `VideoFxProcessor` will not instantiate and will be unsupported. In that case, the following (or similar) error appears in the browser console:

```
Refused to connect to
'https://static.sdkassets.chime.aws/ml_media_fx/otherassets/worker.js'
because it violates the document's content security policy.
```

Using the cross-origin opener policy

To limit memory usage, the module prefers to use a `SharedArrayBuffer` for processing. However, this requires that you carefully configure web security. You must set the following headers when serving your application HTML:

```
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Embedder-Policy: require-corp
```

The server must set these because they have no meta-tag equivalents. If you don't set these headers, the background filters may use slightly more RAM.

Background filters can be CPU-intensive and GPU-intensive. Some mobile devices and lower-specification laptop or desktop computers may not have the power to run the filters along with multiple video streams.

Adding background filters to your application

The process of adding background filters follows these broad steps:

- Check for supported browsers.
- Create a `VideoFxConfig` object with the configuration you want to use.
- Use the configuration object to create a `VideoFxProcessor` object.
- Include the `VideoFxProcessor` object in a `VideoTransformDevice` .
- Use the `VideoTransformDevice` to start the video input.

> **Note**
>
> To complete those steps, you must first:
>
> - Create a `Logger`.
> - Choose a video device of class `MediaDeviceInfo`.
> - Successfully join a `MeetingSession`.

Steps in the following sections explain how to complete the process.

Topics

- Tuning resource utilization

Checking for support before offering a filter

The Amazon Chime SDK provides an asynchronous static method that checks for supported browsers and attempts to download the required assets. However, it does not check for device performance. As a best practice, always ensure the users' browsers and devices can support the filters before you offer the filters.

```
import {
    VideoFxProcessor
} from 'amazon-chime-sdk-js';

if (!await VideoFxProcessor.isSupported(logger)) {
    // logger is optional for isSupported
}
```

Creating a VideoFxConfig object

You can define configurations for `backgroundBlur` and `backgroundReplacement` in the same object. However, you can't set `isEnabled` to `true` for both filters at the same time. That's an invalid configuration.

The `VideoFxConfig` class does no validation of its own. Validation occurs in the next step.

The following example shows a valid `VideoFxConfig`.

```
const videoFxConfig: VideoFxConfig = {
    backgroundBlur: {
        isEnabled: false,
        strength: 'medium'
    },
    backgroundReplacement: {
        isEnabled: false,
        backgroundImageURL: 'space.jpg',
        defaultColor: undefined,
    }
}
```

The following tables list the `VideoFxProcessor` properties that you can specify in the `VideoFxConfig` object.

Background blur filter properties

Property	Type	Description
`isEnabled`	`boolean`	When `true`, the filter blurs the background.
`strength`	`string`	Determines the extent of blurring. Valid values: `low` | `medium` | `high`.

Background replacement filter properties

Property	Type	Description
`isEnabled`	`boolean`	When `true`, the filter replaces the background\.
`backgroundImageURL`	`string`	The URL of the background image. The filter resizes the image dynamically to the dimensions of the current screen. You can use a string such as `https://...` or a data URL such as `data:image/jpeg;base64` .
`defaultColor`	`string`	A hex color string such as `000000` or `FFFFFF`, or a string such as `black` or `white`. If you don't specify an image URL, the processor uses the `defaultColor` as the background. If you don't specify a `defaultColor` , the processor defaults to black.

Creating a VideoFxProcessor object

When creating the `VideoFxProcessor` object, AWS servers download the runtime assets, or a browser cache loads the assets. If network or CSP configurations prevent access to the assets, the `VideoFx.create` operation throws an exception. The resulting VideoFxProcessor is configured as a no-op processor, which won't affect the video stream.

```
let videoFxProcessor: VideoFxProcessor | undefined = undefined;
try {
  videoFxProcessor = await VideoFxProcessor.create(logger, videoFxConfig);
} catch (error) {
  logger.warn(error.toString());
}
```

`VideoFxProcessor.create` also attempts to load the image from `backgroundReplacement.backgroundImageURL`. If the image fails to load, the processor throws an exception. The processor also throws exceptions for other reasons, such as invalid configurations, unsupported browsers, or underpowered hardware.

Configuring the VideoFxProcessor object

The following tables list the `VideoFxProcessor` properties that you can configure. The example below the tables shows a typical runtime configuration.

Background blur

Background blur takes the following properties:

Property	Type	Description
`isEnabled`	`boolean`	When `true`, the filter blurs the background.
`strength`	`string`	Determines the extent of blurring. Valid values: `low` | `medium` | `high` .

Background replacement

Background replacement takes the following parameters:

Property	Type	Description
`isEnabled`	`boolean`	When `true`, the filter replaces the background.
`backgroundImageURL`	`string`	The URL of the background image. The filter resizes the image dynamically to the dimensions of the current screen. You can use a string such as `https://...` or a data URL such as `data:image/jpeg;base64` .
`defaultColor`	`string`	A hex color string such as `000000` or `FFFFFF`, or a string such as `black` or `white`. If you don't specify an image URL, the processor uses the `defaultColor` as the background. If you don't specify a `defaultColor` the processor defaults to black.

Changing a configuration at runtime

You can change a `VideoFxProcessor` configuration at runtime by using the `videoFxProcessor.setEffectConfig` parameter. The following example shows how to enable background replacement and disable background blur.

> **Note**
>
> You can only specify one type of background replacement at a time. Specify a value for `backgroundImageURL` or `defaultColor`, but not both.

```
videoFxConfig.backgroundBlur.isEnabled = false;
videoFxConfig.backgroundReplacement.isEnabled = true;
try {
  await videoFxProcessor.setEffectConfig(videoFxConfig);
} catch(error) {
  logger.error(error.toString())
}
```

If `setEffectConfig` throws an exception, the previous configuration remains in effect. `setEffectConfig` throws exceptions under conditions similar to those that cause `VideoFxProcessor.create` to throw exceptions.

The following example shows how to change a background image while the video runs.

```
videoFxConfig.backgroundReplacement.backgroundImageURL = "https://my-domain.com/my-
other-image.jpg";
try {
  await videoFxProcessor.setEffectConfig(videoFxConfig);
} catch(error) {
  logger.error(error.toString())
}
```

Creating the VideoTransformDevice object

The following example shows how to create a `VideoTransformDevice` object that contains the `VideoFxProcessor`.

```
// assuming that logger and videoInputDevice have already been set
const videoTransformDevice = new DefaultVideoTransformDevice(
  logger,
  videoInputDevice,
  [videoFxProcessor]
);
```

Starting video input

The following example shows how to use the `VideoTransformDevice` object to start video input.

```
// assuming that meetingSession has already been created
await meetingSession.audioVideo.startVideoInput(videoTransformDevice);
meetingSession.audioVideo.start();
```

```
meetingSession.audioVideo.startLocalVideoTile();
```

Tuning resource utilization

When creating the `VideoFxProcessor`, you can supply the optional `processingBudgetPerFrame` parameter and control the amount of CPU and GPU that the filters use.

```
let videoFxProcessor: VideoFxProcessor | undefined = undefined;
const processingBudgetPerFrame = 50;
try {
  videoFxProcessor = await VideoFxProcessor.create(logger, videoFxConfig,
 processingBudgetPerFrame);
} catch (error) {
  logger.warn(error.toString());
}
```

The `VideoFxProcessor` requires time to process a frame. The amount of time depends on the device, the browser, and what else is running in the browser or on the device. The processor uses the concept of a *budget* to target the amount of time used to process and render each frame.

Processing time is in milliseconds. As an example of how to use a budget, 1 second has 1000ms. Targeting 15 frames per second of video capture results in a total budget of 1000ms/15fps = 66ms. You can set a budget of 50% of that, or 33ms, by supplying the value `50` in the `processingBudgetPerFrame` parameter, as shown in the example above.

The `VideoFxProcessor` then tries to process the frames within the budget specified. If processing runs over budget, the processor reduces visual quality to stay within budget. The processor continues to reduce visual quality to a minimum, at which point it stops reducing. This processing duration is measured continually, so if more resources become available, such as another app closing and freeing up CPU, the processor raises visual quality again until it hits the budget, or maximum visual quality is achieved.

If you don't supply a value to `processingBudgetPerFrame`, the `VideoFxProcessor` defaults to `50`.

Example background filter

The following example shows how to implement the filters.

```
import {
```

```
    VideoFxConfig,
    VideoFxTypeConversion,
    VideoTransformDevice,
    DefaultVideoTransformDevice,
    Logger,
    VideoFxProcessor,
    MeetingSession
} from 'amazon-chime-sdk-js';

let videoTransformDevice: VideoTransformDevice | undefined = undefined;
let videoFxProcessor: VideoFxProcessor | undefined = undefined;

const videoFxConfig: VideoFxConfig = {
    backgroundBlur: {
        isEnabled: false,
        strength: "medium"
    },
    backgroundReplacement: {
        isEnabled: false,
        backgroundImageURL: 'space.jpg',
        defaultColor: undefined,
    }
}

export const addEffectsToMeeting = async (videoInputDevice: MediaDeviceInfo,
 meetingSession: MeetingSession, logger: Logger): Promise<void> => {
    try {
        videoFxProcessor = await VideoFxProcessor.create(logger, videoFxConfig);
    } catch (error) {
        logger.error(error.toString());
        return;
    }

    videoTransformDevice = new DefaultVideoTransformDevice(
        logger,
        videoInputDevice,
        [videoFxProcessor]
    );

    await meetingSession.audioVideo.startVideoInput(videoTransformDevice);
}

export const enableReplacement = async (logger: Logger) => {
    videoFxConfig.backgroundBlur.isEnabled = false;
```

```
    videoFxConfig.backgroundReplacement.isEnabled = true;
    await updateVideoFxConfig(videoFxConfig, logger);
}

export const enableBlur = async (logger: Logger) => {
    videoFxConfig.backgroundReplacement.isEnabled = false;
    videoFxConfig.backgroundBlur.isEnabled = true;
    await updateVideoFxConfig(videoFxConfig, logger);
}

export const pauseEffects = async (logger: Logger) => {
    videoFxConfig.backgroundReplacement.isEnabled = false;
    videoFxConfig.backgroundBlur.isEnabled = false;
    await updateVideoFxConfig(videoFxConfig, logger);

}

export const setReplacementImage = async (newImageUrl: string, logger: Logger) => {
    videoFxConfig.backgroundReplacement.backgroundImageURL = newImageUrl;
    videoFxConfig.backgroundReplacement.defaultColor = undefined;
    await updateVideoFxConfig(videoFxConfig, logger);
}

export const setReplacementDefaultColor = async (newHexColor: string, logger: Logger)
 => {
    videoFxConfig.backgroundReplacement.defaultColor = newHexColor;
    videoFxConfig.backgroundReplacement.backgroundImageURL = undefined;
    await updateVideoFxConfig(videoFxConfig, logger);
}

export const setBlurStrength = async (newStrength: number, logger: Logger) => {
    videoFxConfig.backgroundBlur.strength =
 VideoFxTypeConversion.useBackgroundBlurStrengthType(newStrength);
    await updateVideoFxConfig(videoFxConfig, logger);

}

export const updateVideoFxConfig = async (config: VideoFxConfig, logger: Logger) => {
    try {
        await videoFxProcessor.setEffectConfig(videoFxConfig);
    } catch (error) {
        logger.error(error.toString())
    }
}
```

```
export const turnOffEffects = () => {
    const innerDevice = await videoTransformDevice?.intrinsicDevice();
    await videoTransformDevice?.stop();
    videoTransformDevice = undefined;
    videoFxProcessor = undefined;
    await meetingSession.audioVideo.startVideoInput(innerDevice);
}
```

Using the Amazon Chime SDK client library for Windows

Currently, you'll find the Amazon Chime SDK client library for Windows, written in C++, on GitHub. Go to https://github.com/aws/amazon-chime-sdk-cpp.

Frequently asked questions

The topics in the following sections provide answers to frequently asked questions about the Amazon Chime SDK. Expand the topics to learn more.

Topics

Meeting FAQs

Topics

Attendees

Who can join an Amazon Chime SDK meeting?

Only attendees with a required join token. When you use the CreateAttendee, BatchCreateAttendee, or CreateMeetingWithAttendees APIs, you create join tokens that you pass to clients and enable them to join meetings. The tokens generated by those APIs are authenticated by the service, and that grants permission to join the meeting.

> **Note**
>
> The Amazon Chime SDK doesn't create meeting IDs or join URLs for meeting attendees.

What are the meeting attendee quotas?

Attendee quotas are per meeting. The Amazon Chime SDK supports 250 attendees in a standard session, and 100 attendees in a high-definition session. If you need more attendees, consider using media replication. That allows up to 10,000 attendees after requesting a limit increase through the AWS Support Center console. For more information about media replication, refer to Using media replication, earlier in this guide.

Am I charged if there are no attendees in a meeting?

No. The Amazon Chime SDK only charges you when attendees join a meeting. Also, meetings automatically end 5 minutes after the last active attendee drops or leaves the meeting.

What's the difference between AttendeeDeleted, AttendeeLeft, and AttendeeDropped meeting events?

`AttendeeLeft` is triggered when an attendee decides to leave a meeting. `AttendeeDropped` is triggered when an attendee is disconnected from a meeting, usually because of network issues. `AttendeeDeleted` is triggered when the DeleteAttendee API is called.

`AttendeeLeft` is also triggered when:

- When the `DeleteAttendee` API is called from the server-side meeting handler, along with `AttendeeDeleted`.
- Your client calls the `meetingSession.audioVideo.Stop` API from the client library for JavaScript, the corresponding APIs in the iOS and Android SDKs, or when a meeting ends.

For more information about meeting events, refer to Understanding Amazon Chime SDK meeting lifecycle events, earlier in this guide.

How long do meetings run if attendees join but drop off due to bad connections and no one ends the meeting?

Meetings end automatically when:

- The meeting time exceeds 24 hours.
- The meeting is a replica meeting and the primary meeting ends.
- In a non-replica meeting, no attendees connect for 5 continuous minutes.

How long will the Amazon Chime SDK try to reconnect with an attendee?

By default, the Amazon Chime SDK client library for JavaScript tries to reconnect for two minutes, as specified in `MeetingSessionConfiguration` meeting event. Also, the Amazon Chime SDK sends an `AttendeeDropped` event if the attendee is dropped from the meeting and never reconnects to the session.

For more information about meeting events, see Monitoring, later in this FAQ.

Security and encryption

Does the Amazon Chime SDK support end-to-end AWS 256-bit encryption?

Yes. All media is encrypted in-transit and flows through the service. Media is encrypted between the clients and the specific media instance hosting the meeting. The media instance decrypts audio for mixing, then encrypts the mixed audio for transmission to the client. If media is being recorded via media capture, media is encrypted between the media instance and the capture instance.

Audio/video

Topics

General

Does the Amazon Chime SDK pause video when it's backgrounded?

No. However, video streams may be paused when bandwidth is constrained.

How do you prioritize video streams and tiles during meetings? Can you stop a specific video stream during a meeting?

You can programmatically control the video streams that each client subscribes to. This allows you to implement logic such as “presenter always visible” or “meeting host always visible” in a paginated display. If a client is resource constrained, you can turn off the lowest priority streams. For more information, refer to User Guide for Priority-based Downlink Policy on GitHub.

Codecs and simulcast

Which video and audio codecs do you support?

Video codecs

H.264, VP8, VP9, and AV1.

Audio codecs

Opus, 16 kHz, 48 kHz, and 48 kHz stereo.

How does the Amazon Chime SDK support multiple resolutions?

We support video simulcast with VP8 and H.264, and scalable video encoding with VP9. The Amazon Chime SDK client library for JavaScript allows you to specify codec preferences for sending video. The iOS and Android client libraries automatically select a codec for you, based on a device’s capabilities. For more information about video codecs, refer to Configuring video codecs, earlier in this guide. For more information about adaptive simulcast and different policies, refer to Video Simulcast on GitHub.

Echo reduction

Can I selectively apply echo reduction to specific attendees in a meeting?

No. Echo Reduction is enabled at the meeting level for all attendees when you call the CreateMeeting or CreateMeetingWithAttendees APIs. For more information about using echo reduction, refer to Adding Echo Reduction to your application, and Enabling Voice Focus with Echo Reduction, both on GitHub.

Noise suppression

Which noise suppression provider does the Amazon Chime SDK use?

We use Amazon Voice Focus, a noise suppression technology built by AWS. To learn more about Amazon Voice Focus, refer to Understanding VoiceFocus, and Configuring for Amazon Voice Focus, earlier in this guide.

Who can turn noise suppression on and off?

Depending on how you code your solution, meeting attendees usually turn noise suppression on or off. The Amazon Chime SDK client libraries expose programmatic controls for noise suppression, and you choose how to implement them. For example, you can provide a toggle button, or a setting at the app level for controlling noise suppression. For more information, refer to Amazon Voice Focus on GitHub.

Background blur

How much CPU does background blur use?

We have v1 and v2 algorithms. The v1 algorithm has four options based on CPU utilizations (10% to 40%). The v2 algorithm effectively takes an amount of blur to be high, medium or low. The JavaScript browser based demo on GitHub provides a working example.

Screen sharing

What is the screen sharing resolution in web clients?

For a JavaScript client, the browser provides the screen frames for the client library. The resolution is the native resolution of the shared screen, capped by the maximum resolution supported by the meeting. You can set frame rates, but remember that higher frame rates increase CPU loads.

You can also choose a codec for sharing. In standard definition meeting, resolution is 1080p and encoded at 1.5 Mbps. In high definition meetings, resolution is 4K encoded at 2.5 Mbps.

Why can't I share my screen on a mobile device browser?

Mobile device browsers don't support screen capture or screen share. You need to use the iOS or Android SDKs on GitHub to develop an app that supports screen sharing. For more information, refer to the following topics on GitHub:

- Content Share (JavaScript).

- Content Share (iOS).
- Content Share (Android)

Live transcription

How can I redact PII from transcriptions?

You use Amazon Transcribe to redact PII. When you use the StartMeetingTranscription API to transcribe a meeting, you can specify the content redaction type and the different PII entities to redact.

> **Note**
>
> Due to the predictive nature of machine learning, Amazon Transcribe may not identify and remove all instances of sensitive data, and it may not comply with medical privacy laws, such as the U.S. Health Insurance Portability and Accountability Act of 1996 (HIPAA). For more information, refer to Redacting or identifying personally identifiable information, in the *Amazon Transcribe Developer Guide*.

Can I track when transcription starts or ends during a meeting?

Yes. If you subscribe to `transcribeEvent`, every client receives that event, and you can display it in your client to end users. For more information, refer to Understanding transcription events, earlier in this guide.

The following example shows one way to subscribe to `transcribeEvent`.

```
useEffect(() => {
        if (audioVideo) {

 audioVideo.transcriptionController?.subscribeToTranscriptEvent((transcriptEvent) => {
                setTranscripts(transcriptEvent);
            });
        }
    }, [audioVideo]);
```

For more information about using `transcribeEvent`, refer to Understanding transcription events, earlier in this guide.

How do I filter out profanity?

You use Amazon Transcribe to create custom vocabularies and vocabulary filters, and when you call the StartMeetingTranscription API, you provide `VocabularyFilterName` and `VocabularyFilterMethod` values to mask unwanted words. For more information, refer to Custom vocabularies and Creating a vocabulary filter in the *Amazon Transcribe Developer Guide*.

Which languages does live transcription support for meetings?

For real-time live transcription, Amazon Transcribe supports:

- Chinese Simplified (zh-CN)
- English (Australian (en-AU)
- British (en-GB)
- US (en-US))
- French (France(fr-FR) and Canadian (fr-CA))
- German (de-DE)
- Hindi (hi-IN)
- Italian (it-IT)
- Japanese (jp-JP)
- Korean (ko-KR)
- Portuguese (Brazilian (pt-BR)
- Spanish (US (es-US))
- Thai (th-TH)

For more information about the languages available for transcription in real time or batch, refer to Supported languages and language-specific features, in the *Amazon Transcribe Developer Guide*.

Service quotas

I updated a quota in US-EAST-1 (N Virginia). Does the update only apply to the US-EAST endpoint?

Yes. Service quotas are applied per API endpoint. Switching to a different API endpoint applies the default limits.

Namespace migration

Where can I find information on migrating from the chime namespace to the chimesdk namespace?

Refer to the following topics in this guide:

- Migrating from the Amazon Chime namespace.
- Migrating to the Amazon Chime SDK meetings namespace.
- Migrating to the Amazon Chime SDK Identity namespace.
- Migrating to the Amazon Chime SDK Voice namespace.

Are CloudWatch events for Amazon Chime SDK only available on dedicated endpoints and namespaces?

Yes. To use the events, you must migrate from the `chime` namespace to the `chimesdk` namespace. For more information, refer to the following topics in this guide:

- Migrating from the Amazon Chime namespace.
- Migrating to the Amazon Chime SDK meetings namespace.
- Migrating to the Amazon Chime SDK Identity namespace.
- Migrating to the Amazon Chime SDK Voice namespace.

Monitoring

How do you track meeting data such as dates, times, call durations, and attendees?

We send meeting and attendee events via Amazon EventBridge, Amazon SNS, or Amazon SQS. The events contain information such as meeting start and stop times, and attendee join, drop, and leave actions. For more information about meeting events and how to use them, refer to the following topics:

- Understanding Amazon Chime SDK event notifications, earlier in this guide.
- The Server-side logging and monitoring of Amazon Chime SDK events blog post.
- The Monitoring and troubleshooting with Amazon Chime SDK meeting events blog post.
- Meeting Events on GitHub.

Which CloudWatch metrics are available?

The metrics include `AttendeeAuthorizationSuccess`, `AttendeeAuthorizationErrors`, and `AttendeeAudioDrops`. To learn more about the metrics, refer to Understanding Amazon CloudWatch metrics for the Amazon Chime SDK, earlier in this guide.

How do I build a dashboard for logging and monitoring?

The Amazon Chime SDK generates meeting events based on different states of the components in your client application, such as audio, video, screen sharing, or attendee activities. You can write those events to CloudWatch logs, then build a dashboard on those logs. You can include different events, error messages, and status codes to help gain insights from the data.

The Amazon Chime SDK also integrates with Amazon EventBridge, Amazon SQS, and Amazon SNS for tracking server-side events such as requests to create or delete meetings, attendees, or media pipelines. You can configure rules to filter for events you are interested in and write the events to CloudWatch logs.

To learn more about meeting events and using them to create dashboards, refer to:

- Meeting Events on GitHub.
- Understanding Amazon Chime SDK meeting lifecycle events, earlier in this guide.
- Understanding Amazon CloudWatch metrics for the Amazon Chime SDK, earlier in this guide.
- Understanding Amazon Chime SDK event notifications, earlier in this guide.

To try it, follow the instructions in these blog posts:

- Server-side logging and monitoring of Amazon Chime SDK events.
- Monitoring and troubleshooting with Amazon Chime SDK meeting events.

How can I monitor whether meetings end automatically or when the DeleteMeeting API is called?

Both ways of ending a meeting trigger the `MeetingFailed` event. If you don't have a Cloud Trail or EventBridge entry for the DeleteMeeting API, you can assume that the meeting ended automatically.

Logging

How do I enable WebRTC debug logging on Google Chrome?

Run the following command and flags: `chrome –enable-logging –vmodule=*/webrtc/*=1`. This turns on `IFO` and `VERBOSE` logging for WebRTC. The resulting log is named `chrome_debug.log` and saved in the Chrome user data directory.

How to enable WebRTC debug logging for Safari on macOS?

Follow these steps:

1. In Safari, select **Settings**.
2. Choose **Advanced options**, then choose **Show features for web developers**.

   The **Develop** menu appears in the browser.
3. On the **Develop** menu, choose **Show JavaScript Console**.
4. In the JavaScript console, choose **Settings**, then enable WebRTC logging. You can choose basic or verbose logging as needed.

Error messages

How do I troubleshoot the “Session stopped - reason - ICEGatheringTimeout Workground” error?

Do the following:

- Ensure that egress for UDP port 3478 on IP range 99.77.128.0/18 is enabled. For more information, refer to Configuring your network for the Amazon Chime SDK, earlier in this guide.
- Ensure that an anti-virus browser extension isn't preventing resources from loading. UDP 3478 is for TURN and needs to be unblocked on the end user side, either on local computer firewalls or on the corporate network firewall.
- Connection retry falls back to TLS over port 443, so ensure that the domain or subnet are not blocked.

What does the “Error: Invalid capture pipeline ARN” message mean?

This error typically occurs when the service can’t resolve a media pipeline ARN. Ensure the ARN belongs to a media pipeline and not to a meeting. `MediaPipelineArn` is part of the CreateMediaCapturePipeline API response.

What does the "AudioJoinedFromAnotherDevice" error mean, and how do I avoid it?

This error is raised when the same attendee joins from two devices. The error is returned in the `meetingErrorMessage` attribute of a `meetingFailed` event. To avoid this, ensure that each attendee has a unique `ExternalUserId`, and make sure that you do not use the same attendee response from the CreateAttendee, BatchCreateAttendee, or CreateMeetingWithAttendees APIs in two or more meetings simultaneously.

How do I resolve “Forbidden: Not authorized to call Chime SDK with Account Id *111122223333*”?

You're calling a deprecated Amazon Chime API. To solve the problem, migrate to the Amazon Chime SDK namespace. For more information, refer to the following topics earlier in this guide:

- Migrating from the Amazon Chime namespace.
- Migrating to the Amazon Chime SDK meetings namespace.
- Migrating to the Amazon Chime SDK Identity namespace.
- Migrating to the Amazon Chime SDK Voice namespace.

How do I resolve “Forbidden: Account Id *111122223333* is not authorized to call deprecated Amazon Chime SDK API on the chime endpoint”?

You're calling a deprecated Amazon Chime API. To solve the problem, migrate to the Amazon Chime SDK namespace. For more information, refer to the following topics earlier in this guide:

- Migrating from the Amazon Chime namespace.
- Migrating to the Amazon Chime SDK meetings namespace.
- Migrating to the Amazon Chime SDK Identity namespace.
- Migrating to the Amazon Chime SDK Voice namespace.

For media pipelines, how do I troubleshoot "Runtime error: problem contacting Chime: Client request token exists without active resources, please re-generate client request token"?

The client request token is a unique identifier that makes API requests idempotent. This error occurs when the token is associated with an inactive media pipeline. To fix the issue, generate a new unique token and send it with the API request.

Media pipeline FAQs

Which format does media capture use for 5 second segments?

Media capture uses the MP4 format. This includes 5 second segments and combined recording and composited files.

How do I delete attendees created by media capture pipelines?

To delete media capture attendees, you can end the pipeline or call the DeleteMediaCapturePipeline API.

Does recording happen in the cloud or locally?

Media capture pipelines record directly into your Amazon S3 bucket. Media capture doesn't put any bandwidth or connectivity requirements on clients.

Where can we create media capture in relation to meetings?

To choose a Region for creating media captures, first choose an API endpoint from the available meeting control plane Regions. Next, create the meeting and media capture pipeline in that Region. Media capture can write to an Amazon S3 bucket in your account in any Amazon Chime SDK media Region. For more information about the available Regions and endpoints, the media pipeline control plane, and media Regions, refer to Available AWS Regions for the Amazon Chime SDK service earlier in this guide, and Amazon Chime SDK endpoints and quotas in the *AWS Reference guide*.

Does media capture record all 250 attendee videos in a meeting?

No. A pipeline only captures the first 25 video streams.

Can I stop recording while a meeting continues?

You can create a mechanism to call the DeleteMediaCapturePipeline API after a given number of minutes. For example, you can create a step function that starts when media capture begins and has a predetermined wait time.

Can I stop meetings while recording is on?

You can call the DeleteMediaCapturePipeline API to end the recordings, or you can call the DeleteMeeting API when the meeting is scheduled to end. During a meeting, if the media capture attendee is the only attendee left, the meeting automatically ends after 5 minutes.

PSTN audio FAQs

Can you use PSTN audio to route inbound calls from non-US numbers to Voice Connectors?

No. You cannot route a non-US number to a Voice Connector by using PSTN Audio.

When attendees connect via PSTN audio, can you move them from the current meeting to a new meeting?

Yes. First call the Hangup action for the leg connected to the meeting. That disconnects the attendee from the meeting without terminating the inbound call. Then call the JoinChimeMeeting action to join the attendees to the new meeting.

Document history

The following table describes important changes to the *Amazon Chime Developer Guide*, beginning in September 2019. For notifications about updates to this documentation, you can subscribe to an RSS feed.

Change	Description	Date
Alexa in-skill calling removed	Due to changes by the Amazon Alexa team, you can no longer add Alexa calls to SIP media applications. For more information, refer to the Alexa Smart Properties page.	April 1, 2024
New meeting Regions	Developers can now use several new meeting Regions. For more information, refer to Available Regions in this guide, and Amazon Chime SDK endpoints and quotas, in the *AWS General Reference*.	September 25, 2023
Voice enhancement	Developers can now enable call recording and store the recorded calls in an Amazon S3 bucket. For more informati on, refer to Understanding voice enhancement in this guide.	August 31, 2023
Updated regions	Developers who use the Amazon Chime SDK can now use more regions. For more information, refer to Available regions.	August 29, 2023

Call analytics and voice analytics	Developers can now add low-code analytics and transcrip tion capabilities to their solutions. For more informati on, refer to Using Amazon Chime SDK call analytics in this guide.	March 27, 2023
Client library for Windows	Developers can now use the Amazon Chime SDK client library for Windows, written in C++. For more information, refer to Amazon Chime SDK client library for Windows in this guide.	February 2, 2023
Updated regions	Developers who use the Amazon Chime SDK can now use more regions. For more information, refer to Available regions.	November 18, 2022
C++ client library on GitHub	Developers who use Amazon Chime SDK Meetings can now integrate with a C++ signaling client library on GitHub. For more informati on, refer to Integrating with a client library.	August 19, 2022

Media pipelines	Developers who use Amazon Chime SDK Meetings now create *media pipelines*. In turn, media pipelines consist of media capture pipelines , media concatenation pipelines, and live connector pipelines. For more informati on, refer to Creating Amazon Chime SDK media pipelines.	August 18, 2022
Elastic channels	Developers who use Amazon Chime SDK Messaging can now use elastic channels in their chat solutions. Elastic channels can host up to 1-million users. For more information, refer to Using elastic channels to host live events.	August 12, 2022
Emergency 911 address validation	Developers who use Amazon Chime SDK Meetings can programmatically validate the addresses that emergency calls originate from. For more information, refer to ValidateE911Address in the *Amazon Chime API Reference* . and Validating addresses for emergency calls in the *Amazon Chime SDK Administr ation Guide*.	August 11, 2022

Meeting tags update	Developers can now use meeting tags in the `Chime` and `ChimeSDKMeetings` namespaces. For more information, see Migrating to the Amazon Chime SDK Meetings namespace.	August 4, 2022
CallAndBridge action for Voice Connectors and Voice Connector groups	Developers who use the Amazon Chime SDK Audio Service can use the `CallAndBridge` action to place outbound calls to SIP trunks configured as Voice Connectors or Voice Connector Groups. For more information, refer to CallAndBridge.	July 14, 2022
AppKeys and TenantIDs	Developers who use Amazon Chime SDK Meetings can now use `AppKeys` and `TenantIDs` to control access to WebRTC media sessions from their customer's networks. For more information, refer to Using AppKeys and Tenant IDs.	July 7, 2022

Connect API	Developers who use Amazon Chime SDK Messaging can now use WebSockets to connect to back-end servers and receive messages for an `AppInstanceUser` . For more information, refer to Using the connect API and Using WebSockets to receive messages.	June 6, 2022
Attendee capabilities	Developers can now use capabilities to control attendees' access to audio, video, and content during Amazon Chime SDK meetings. For more information, refer to AttendeeCapabilities in the *Amazon Chime SDK API Reference*.	June 2, 2022
Amazon CloudWatch metrics	Developers can now take advantage of service and usage metrics that the Amazon Chime SDK publishes to CloudWatch. The metrics enable you to use CloudWatch graphs and dashboards to monitor how you consume Amazon Chime SDK services. For more information see Amazon CloudWatch metrics.	June 1, 2022

Echo reduction	Developers can now implement echo reduction, which helps keep echoes—sounds from a user's loudspeaker that get picked up by their microphone—from circulating back into meeting audio and bringing discussions to a standstill. For more information, refer to Using echo reduction.	November 23, 2021
Call recording	Developers can now implement audio recording on one or more legs of an Amazon Chime SDK SIP media application call. For more information, refer to Using call recording. In addition, the `RecordAudio` action now includes new parameters, including `SilenceDurationInSeconds` and `RecordingTerminatorUsed`. For more information, refer to RecordAudio.	October 28, 2021
Background blur	Developers can now add background blur to their Amazon Chime SDK applications. For more information, refer to Using background blur.	October 21, 2021

Updated IAM policy	Developers now have an updated IAM policy that supports live transcription for Amazon Chime SDK meetings. For more information, refer to Creating IAM users or roles with the Chime SDK policy.	September 22, 2021
SIP headers	Developers can now send and receive a User-To-User header, a Diversion header, and custom SIP headers in their AWS Lambda functions. For more information, refer to Using SIP headers.	September 13, 2021
Amazon Chime SDK meetings	Developers can now use Amazon Chime SDK live transcription. For more information, refer to Using Amazon Chime SDK live transcription.	August 11, 2021
Amazon Chime SDK meetings	Developers can now create media pipelines. For more information, refer to Creating Amazon Chime SDK media capture pipelines.	July 7, 2021
Using SIP media applications with AWS Lambda functions	Title changed from "Lambda functions SDK" and content in all topics revised for accuracy. CallAndBridge section added. For more information, refer to Using the PSTN audio service and CallAndBridge.	June 17, 2021

Lambda functions SDK	Developers can build custom Lambda functions for use in the Amazon Chime SDK SIP media applications created by Amazon Chime SDK administrators. For more information, refer to Using the PSTN audio service in the *Amazon Chime Developer Guide*.	November 17, 2020
JavaScript SDK	Developers can use JavaScript to build Amazon Chime SDK applications. For more information, refer to Using the Amazon Chime SDK for JavaScript in the *Amazon Chime Developer Guide*.	November 17, 2020
Android and iOS client libraries	Developers can find the client libraries for Android, iOS, and Windows in less time and with fewer clicks. For more information, refer to Using the Amazon Chime SDK client library for Android and Using the Amazon Chime SDK client library for iOS in the *Amazon Chime Developer Guide*.	November 17, 2020
Proxy phone sessions	Developers can create proxy phone sessions for use with Amazon Chime SDK Voice Connectors. For more information, refer to Using the Amazon Chime SDK for JavaScript in the *Amazon Chime Developer Guide*.	April 7, 2020

Amazon Chime SDK content sharing	The Amazon Chime SDK supports content sharing. For more information, refer to Amazon Chime SDK architecture in the *Amazon Chime Developer Guide*.	March 31, 2020
Amazon Chime SDKs for Android and iOS	The Amazon Chime SDKs for Android and iOS is released. For more information, refer to Integrating with a client library in the *Amazon Chime Developer Guide*.	March 24, 2020
Amazon Chime SDK	The Amazon Chime SDK is released. For more information, refer to Using the Amazon Chime SDK in the *Amazon Chime Developer Guide*.	November 20, 2019
Amazon Chime Developer Guide	The *Amazon Chime Developer Guide* is released.	September 11, 2019