aws

Chat Messaging API Reference

Amazon IVS

Amazon IVS: Chat Messaging API Reference

Table of Contents

Welcome

The messaging API for Amazon IVS Chat is designed to be used by your client applications (e.g., web-browser and Android/iOS apps) to interact with chat rooms alongside your live video. This is a WebSockets API using a token-based authentication scheme. WebSockets are widely supported by browsers and mobile client libraries to facilitate fully bidirectional communication between client and server.

Clients connect to a room over WebSockets (WSS) using secure access tokens created by your application with the CreateChatToken endpoint. Clients initiate a handshake to a WebSocket endpoint in the same region as the room they want to connect to. Clients make messaging requests by publishing JSON payloads over the WebSocket connection. Messaging requests are asynchronous. Success should be assumed if an error is not returned.

The API is an AWS regional service. For a list of supported regions, see the Amazon IVS Chat information on the Amazon IVS page in the *AWS General Reference*.

There are two sets of messaging operations:

- **Publish operations** are JSON payloads that clients can send to the Amazon IVS Chat service:
  - DeleteMessage — Instructs other clients to delete a message.
  - DisconnectUser — Disconnects another viewer from the Chat room. Use this in conjunction with the CreateChatToken flow to implement time-based or permanent user bans. For example, to permanently ban a user (viewer), precede this operation with a call to your application service, to prevent the application service from creating another token to the same room. To lift the ban (time- or logic-based), simply resume calling CreateChatToken.
  - SendMessage — Sends a message to participants of the room.
- **Subscribe operations** are JSON payloads that clients will receive from the Amazon IVS Chat service. Error messages are pushed to clients if an action fails.
  - Event — Events are messages generated by client actions or operations; e.g., join or leave notifications. Events may be generated by Amazon IVS Chat or your application. Applications should use the SendEvent HTTP endpoint to push event messages to the room.
  - Message — Messages from other clients that are sent to the room.

All the above are *messaging* operations. There is a separate API (all HTTP) for managing Chat resources; see the Amazon IVS Chat API Reference.

Connecting to a Room

1. When a viewer wants to connect to a room, the browser or client application requests a chat token from your application.
2. Your application determines whether the viewer should be authorized to chat. If yes, your application calls `CreateChatToken` to create an encrypted token. The token specifies the room to connect to, attributes for the user, and capabilities the connection will have in the room.
3. The token is returned to the client application. The token must be used to connect to a room within a brief period, and the token can only be used once.
4. The client application connects to the WebSocket endpoint, passing the token as an HTTP header. Tokens must be presented during the WebSocket handshake via the `sec-websocket-protocol` header.
5. During the WebSocket handshake, the token is securely decrypted and validated by Amazon IVS Chat.
6. If the handshake is successful, Amazon IVS Chat associates the specified user attributes and capabilities with the WebSocket connection.
7. The client application can begin making messaging requests and will receive subscribe operations from the room.

Actions

The following actions are supported:

- the section called “DeleteMessage (Publish)” — Instructs other clients to delete a message.
- the section called “DisconnectUser (Publish)” — Disconnects another viewer from the Chat room. Use this in conjunction with the CreateChatToken flow to implement time-based or permanent user bans. For example, to permanently ban a user (viewer), precede this operation with a call to your application service, to prevent the application service from creating another token to the same room. To lift the ban (time- or logic-based), simply resume calling CreateChatToken.
- the section called “SendMessage (Publish)” — Sends a message to participants of the room.
- the section called “Event (Subscribe)” — Events are messages generated by client actions or operations; e.g., join or leave notifications. Events may be generated by Amazon IVS Chat or your application. Use SendEvent (see the Amazon IVS Chat API Reference) to push event messages to the room.
- the section called “Message (Subscribe)” — Messages from other clients that are sent to the room.

DeleteMessage (Publish)

Instructs other clients to delete a message.

Required Capability

`DELETE_MESSAGE`

Format

```
{
  "Action": "DELETE_MESSAGE",
  "Id": "string",
  "Reason": "string",
  "RequestId": "string"
}
```

Fields

Field	Required	Description
`Action`	Yes	`DELETE_MESSAGE`
`Id`	Yes	ID of the message to be deleted. This is the Id field in the received message (see the section called "Message (Subscribe)").
`Reason`	No	Reason for deleting the message.
`RequestId`	No	An optional identifier specified by your application for tracking purposes. If specified, this appears in corresponding subscribe operations.

DisconnectUser (Publish)

Disconnects another viewer from the Chat room. Use this in conjunction with the CreateChatToken flow to implement time-based or permanent user bans. For example, to permanently ban a user (viewer), precede this operation with a call to your application service, to prevent the application service from creating another token to the same room. To lift the ban (time- or logic-based), simply resume calling CreateChatToken.

Required Capability

`DISCONNECT_USER`

Format

```
{
  "Action": "DISCONNECT_USER",
  "RequestId": "string",
  "Reason": "string",
  "UserId": "string"
}
```

Fields

Field	Required	Description
`Action`	Yes	`DISCONNECT_USER`
`RequestId`	No	An identifier optionally specified by your application for tracking purposes. If specified, this appears in corresponding subscribe operations.
`Reason`	No	Reason for disconnecting the user.
`UserId`	Yes	User ID of the user(s) to disconnect from the room. If multiple connections share this ID, all are disconnected.

SendMessage (Publish)

Sends a message to participants of the room.

Required Capability

`SEND_MESSAGE`

Format

```
{
  "Action": "SEND_MESSAGE",
  "Attributes": {
    "user_id": "string",
    "display_name": "string"
  },
  "Content": "string",
  "RequestId": "string"
}
```

Fields

Field	Required	Description
`Action`	Yes	`SEND_MESSAGE`
`Attributes`	No	Details of the event; e.g., `user_id`, `display_name` , and/or anything you want.
`Content`	Yes	Message to send. The character length of this field must be shorter than the room's configured `maximumMessageLength` . Maximum: 500 unicode code points.
`RequestId`	No	An identifier optionally specified by your applicati on for tracking purposes. If specified, this appears in corresponding subscribe operations.

Event (Subscribe)

Events are messages generated by client actions or operations; e.g., join or leave notifications. Events may be generated by Amazon IVS Chat or your application. Use SendEvent (see the Amazon IVS Chat API Reference) to push event messages to the room.

Format

```
{
  "Attributes": { "string": "string" },
  "EventName": "string",
  "Id": "string",
  "RequestId" : "string",
  "SendTime": "string::date-time",
  "Type": "EVENT"
}
```

Fields

Field	Description
`Attributes`	For application-generated events, this contains details of the event; e.g., `user_id`, `display_name` , and/or anything you want. For the system-generated event (`aws:DELETE_MESSAGE`), this contains three fields: • `MessageId` — The ID of the message that was deleted. • `MessageID` — Deprecated. Use `MessageId` instead. • `Reason` — An explanation of why the message was deleted.
`EventName`	For application-generated events, this is preferably an enum used by client logic; e.g., `user_joined` . There is one system-generated event. Its `EventName` is `aws:DELETE_MESSAGE` .
`Id`	An identifier generated by Amazon IVS Chat.
`RequestId`	An identifier optionally specified by your application (in the corresponding SendEvent) for tracking purposes.
`SendTime`	Timestamp of when the message was received by Amazon IVS Chat.
`Type`	`EVENT`

Message (Subscribe)

Messages from other clients that are sent to the room.

Format

```
{
  "Attributes": { "string": "string" },
  "Content": "string",
```

```
  "Id": "string",
  "RequestId": "string",
  "Sender": {
    "Attributes": { "string": "string" },
    "UserId": "string"
  },
  "SendTime": "string::date-time",
  "Type": "MESSAGE"
}
```

Fields

Field	Description
`Attributes`	Attributes included in the message.
`Content`	Message received by another user.
`Id`	An identifier generated by Amazon IVS Chat.
`RequestId`	The message identifier optionally specified by your application (in the corresponding the section called "SendMessage (Publish)" operation) for tracking purposes
`Sender`	Information about the sender. This includes two fields: • `Sender.Attributes` is metadata about the sender established during authentication. This can be used to give the client more information about the sender; e.g., avatar URL, badges, font, and color. • `Sender.UserId` is an application-specified identifier of the viewer (end user) who sent this message. This can be used by the client application to refer to the user in either the messaging API or application domains.
`SendTime`	Timestamp of when the message was received by Amazon IVS Chat.
`Type`	`MESSAGE`

Error Messages

WebSocket Errors

Error messages are pushed to clients if an operation fails. Due to the asynchronous nature of the Chat messaging API, WebSocket error messages include an identifier that the client can use to map back to the responsible operation.

Here is a sample WebSocket operation error message:

```
{
  "ErrorCode": 400,
  "ErrorMessage": "Messages must be less than 240 chars",
  "Id": “IG5mcmlvdGlv”,
  "RequestId": "497f6eca-6276-4993-bfeb-53cbbbba6f08",
  "Type": "ERROR"
}
```

Field	Format	Description
ErrorCode	int	Error code (see below).
ErrorMessage	string	Descriptive error message.
Id	string	Identifier of the operation that caused this error. If the original operation had an identifier generated by Amazon IVS Chat, it is used here; otherwise, the Chat service generates an identifier for tracking purposes.
RequestId	string	An identifier optionally specified by your application for tracking purposes.
Type	string	ERROR

Error codes map to HTTP status codes which describe errors that clients may encounter when interacting with the Chat messaging WebSocket operations:

Error	HTTP Status Code	Description
Bad Request	400	The request is malformed or invalid.
Unauthorized	401	The connection has expired or the chat token is missing.
Forbidden	403	The connection's capabilities do not permit this action, or the account is blocked from using chat because the account is pending verification.
Not Found	404	The room cannot be found (does not exist).
Not Acceptable	406	The request was rejected during message review.
Payload Too Large	413	The request contains fields that are too large.
Too Many Requests	429	The application has exceeded rate limits.
Internal Server Error	500	This is returned for a general error.

HTTP Errors

For errors common to all HTTP actions, see Common Errors in the *Amazon IVS API Reference*. For errors specific to Chat messaging, see the following table:

Error	HTTP Status Code	Description
InvalidParameterValue	400	The Chat message exceeds the character limit. See the section called "SendMessage (Publish)".
ResourceNotFound	404	A resource referenced/mutated by the operation does not exist.