Playing Streams

1 Introduction

When users participate in a call or a live broadcast, they need to play the streams published by other participants from ZEGO's cloud servers if they want to see the video of those participants.

2 Procedures

2.1 Listen for and Handle Stream Playing Related Events

Before starting to play streams, you need to register event listeners to listen for and handle stream playing related events, such as a stream is pulled successfully or the stream playing fails due to network problems.

```
zg.on('playerStateUpdate', result => {
//Handle stream playing status update
})

zg.on('playQualityUpdate', (streamID, stats) => {
//Handle stream playing quality update
})
```

It is recommended to handle the following events :

- playerStateUpdate: You can determine whether the stream playing is successful or not based on the status returned. If the stream playing is interrupted due to network problems, notifications will be sent through this callback, and the web client needs to handle it properly by prompting a notice or retry it automatically.
- playQualityUpdate: Stream playing quality update includes information such as the frame rate, bit rate of the video and audio streams, which can be used to determine the audio and video playback quality on the client side.

- Please handle other events as needed according to your business requirements. Please refer to ZEGO Express Web SDK API Reference Section 4 for events related to stream playing.

2.2 Start Playing Streams

Call the API `startPlayingStream` to start pulling a remote stream, and the rendering of the remote stream can be started by assigning the remote stream to the `srcObject` property of a local `video` or `audio` object.

```
// async function and await keyword are syntax introduced by ES2017 standard
// Promise returns the streaming media object asynchronously
const remoteStream = await zg.startPlayingStream(streamID);

//remoteVideo is a local <video> or <audio> object
remoteVideo.srcObject = remoteStream;
```

If you are using the **Vue** framework, you can reder the stream by assigning `stream` to `srcObject.prop`.

```
<video autoplay v-for="vd in streamList" :srcObject.prop="vd.stream"></video>
```

- `streamID` can be obtained from the callback roomStreamUpdate, which should be a string with a maximum length of 256 bytes and contains only numbers, letters, and underscores. It must be globally unique within the entire scope of the AppID.
- `localVideo` is the `<video>` or `<audio>` object for local preview. It is recommended to add `autoply` property to the object.
- For some web browsers, autoplay is permitted only when the `muted` property is enabled. In such cases, it is recommended to prompt users to turn on the audio.
- For the acquisition of `streamID`, the newly pushed or stopped stream before and after the user logs in to the room can be obtained from the stream information update callback roomStreamUpdate callback.

- Because the iOS system restricts the automatic playback strategy of `<video>` (this strategy requires special processing by the business layer), the web terminal under this system cannot automatically play the video, and the user needs to manually trigger the playback event.

2.3 Stop Playing Streams

When a stream is deleted, call the API `stopPlayingStream` to stop playing the stream.

```
zg.stopPlayingStream(streamID)
```

After stream playing is stopped, the related local `<video>` or `<audio>` object also needs to be removed accordingly in your program.

3 API Reference

Method	Description
startPlayingStream	Starts playing a stream.
stopPlayingStream	Stops playing a stream.
Callback	**Description**
playerStateUpdate	The callback triggered when the status of stream playing changes.
playQualityUpdate	The callback triggered every 3 seconds to report the current stream playing quality.
remoteCameraStatusUpdate	The callback triggered when the status of the remote camera changes.
remoteMicStatusUpdate	The callback triggered the status of the remote microphone changes.

4 Related Documents

If users want to have two-way video communication with other participants in a call or live broadcast, they will also need to publish their own video to ZEGO's cloud servers. Use the next/previous buttons below or the side navigation bar on the left to navigate to the related documents.

- QuickStart - Publishing Streams