AKOOL

Streaming Avatar › **Streaming Avatar SDK API Reference**

Streaming Avatar

Streaming Avatar SDK API Reference

Complete API documentation for akool-streaming-avatar-sdk

Overview

This document provides comprehensive API documentation for the **akool-streaming-avatar-sdk** package. The SDK provides a generic JavaScript interface for integrating Agora RTC streaming avatar functionality into any JavaScript application.

Package Links:

- NPM Package
- GitHub Repository
- **Current Version:** 1.0.6

GenericAgoraSDK Class Documentation

Constructor

`new GenericAgoraSDK(options?: { mode?: string; codec?: SDK_CODEC })`

Creates an instance of the Generic Agora SDK.

Parameters:

- `options.mode` (string) - Optional. SDK mode (default: “rtc”)
- `options.codec` (SDK_CODEC) - Optional. Video codec (e.g., “vp8”, “h264”)

Example Usage:

```
import { GenericAgoraSDK } from 'akool-streaming-avatar-sdk';

const agoraSDK = new GenericAgoraSDK({ mode: "rtc", codec: "vp8" });
```

Connection Management

`async joinChannel(credentials: AgoraCredentials): Promise<void>`

Joins an Agora RTC channel using the provided credentials.

Parameters:

- `credentials` (AgoraCredentials) - Agora connection credentials

Example Usage:

```
await agoraSDK.joinChannel({
  agora_app_id: "your-agora-app-id",
  agora_channel: "your-channel-name",
  agora_token: "your-agora-token",
  agora_uid: 12345
});
```

`async leaveChannel(): Promise<void>`

Leaves the current Agora RTC channel.

Example Usage:

```
await agoraSDK.leaveChannel();
```

`async closeStreaming(cb?: () => void): Promise<void>`

Closes all connections and performs cleanup.

Parameters:

- `cb` (function) - Optional callback function to execute after closing

Example Usage:

```
await agoraSDK.closeStreaming(() => {
  console.log("Streaming closed successfully");
});
```

`isConnected(): boolean`

Checks if the SDK is connected to Agora services.

Returns:

- `boolean` - Connection status

Example Usage:

```
const connected = agoraSDK.isConnected();
console.log("Connected:", connected);
```

`isChannelJoined(): boolean`

Checks if the SDK has joined a channel.

Returns:

- `boolean` - Channel join status

Example Usage:

```
const joined = agoraSDK.isChannelJoined();
console.log("Channel joined:", joined);
```

Chat Management

`async joinChat(metadata: Metadata): Promise<void>`

Initializes the avatar chat session with the specified parameters.

Parameters:

`metadata` (Metadata) - Avatar configuration parameters

Example Usage:

```
await agoraSDK.joinChat({
  vid: "voice-id-12345",
  lang: "en",
  mode: 2, // 1 for repeat mode, 2 for dialog mode
  bgurl: "https://example.com/background.jpg"
});
```

`setParameters(metadata: Metadata): void`

Sets avatar parameters. Refer to the avatar parameter documentation for details.

Parameters:

`metadata` (Metadata) - Avatar configuration parameters

Example Usage:

```
agoraSDK.setParameters({
  vid: "new-voice-id",
  lang: "es",
  mode: 1
});
```

`async leaveChat(): Promise<void>`

Leaves the chat session but maintains the channel connection.

Example Usage:

```
await agoraSDK.leaveChat();
```

`async sendMessage(content: string): Promise<void>`

Sends a text message to the avatar.

Parameters:

- `content` (string) - The message content to send

Example Usage:

```
await agoraSDK.sendMessage("Hello, how are you today?");
```

`async interrupt(): Promise<void>`

Interrupts the current avatar response.

Example Usage:

```
await agoraSDK.interrupt();
```

`getMessages(): Message[]`

Returns all chat messages in the current session.

Returns:

- `Message[]` - Array of chat messages

Example Usage:

```
const messages = agoraSDK.getMessages();
console.log("All messages:", messages);
```

`getMessage(messageId: string): Message | undefined`

Returns a specific message by its ID.

Parameters:

- `messageId` (string) - The ID of the message to retrieve

Returns:

- `Message | undefined` - The message object or undefined if not found

Example Usage:

```
const message = agoraSDK.getMessage("msg-123");
if (message) {
  console.log("Message text:", message.text);
}
```

Audio Management

`async toggleMic(): Promise<void>`

Toggles the microphone on or off.

Example Usage:

```
await agoraSDK.toggleMic();
console.log("Microphone enabled:", agoraSDK.isMicEnabled());
```

`isMicEnabled(): boolean`

Checks if the microphone is currently enabled.

Returns:

- `boolean` - Microphone status

Example Usage:

```
const micEnabled = agoraSDK.isMicEnabled();
console.log("Microphone is:", micEnabled ? "on" : "off");
```

Client Access

`getClient(): RTCClient`

Returns the underlying Agora RTC client instance for advanced operations.

Returns:

- `RTCClient` - The Agora RTC client

Example Usage:

```
const client = agoraSDK.getClient();
// Use client for advanced Agora operations
```

Event Handling

`on(events: SDKEvents): void`

Registers event handlers for various SDK events.

Parameters:

- `events` (SDKEvents) - Object containing event handler functions

Example Usage:

```
agoraSDK.on({
  onStreamMessage: (uid, message) => {
    console.log(`Message from ${uid}:`, message);
  },
  onException: (error) => {
    console.error("SDK Exception:", error);
  },
  onMessageReceived: (message) => {
    console.log("New message received:", message);
  },
  onMessageUpdated: (message) => {
    console.log("Message updated:", message);
  },
  onNetworkStatsUpdated: (stats) => {
    console.log("Network stats updated:", stats);
  },
  onTokenWillExpire: () => {
    console.warn("Token will expire in 30 seconds");
  },
  onTokenDidExpire: () => {
    console.error("Token has expired");
  },
  onUserPublished: async (user, mediaType) => {
    if (mediaType === 'video') {
      const remoteTrack = await agoraSDK.getClient().subscribe(user, mec
      remoteTrack?.play('video-container-id');
    } else if (mediaType === 'audio') {
      const remoteTrack = await agoraSDK.getClient().subscribe(user, mec
      remoteTrack?.play();
    }
  },
  onUserUnpublished: (user, mediaType) => {
    console.log(`User ${user.uid} stopped publishing ${mediaType}`);
  }
});
```

Type Definitions

AgoraCredentials

Interface for Agora connection credentials.

```
interface AgoraCredentials {
  agora_app_id: string;     // Agora App ID
  agora_channel: string;    // Channel name
  agora_token: string;      // Agora token
  agora_uid: number;        // User ID
}
```

Metadata

Interface for avatar configuration parameters.

```
interface Metadata {
  vid?: string;      // Voice ID
  vurl?: string;     // Voice URL
  lang?: string;     // Language code (e.g., "en", "es", "fr")
  mode?: number;     // Mode type (1: repeat, 2: dialog)
  bgurl?: string;    // Background image URL
}
```

Message

Interface for chat messages.

```
interface Message {
  id: string;           // Unique message ID
  text: string;         // Message content
  isSentByMe: boolean;  // Whether the message was sent by the current u
}
```

NetworkStats

Interface for network statistics.

```
interface NetworkStats {
  localNetwork: NetworkQuality;        // Local network quality
  remoteNetwork: NetworkQuality;       // Remote network quality
  video: RemoteVideoTrackStats;        // Video track statistics
  audio: RemoteAudioTrackStats;        // Audio track statistics
}
```

SDKEvents

Interface defining all available event handlers.

```
interface SDKEvents {
  onStreamMessage?: (uid: UID, message: StreamMessage) => void;
  onException?: (error: { code: number; msg: string; uid: UID }) => voic
  onNetworkQuality?: (stats: NetworkQuality) => void;
  onUserJoined?: (user: IAgoraRTCRemoteUser) => void;
  onUserLeft?: (user: IAgoraRTCRemoteUser, reason: string) => void;
  onRemoteAudioStats?: (stats: RemoteAudioTrackStats) => void;
  onRemoteVideoStats?: (stats: RemoteVideoTrackStats) => void;
  onTokenWillExpire?: () => void;
  onTokenDidExpire?: () => void;
  onMessageReceived?: (message: Message) => void;
  onMessageUpdated?: (message: Message) => void;
  onNetworkStatsUpdated?: (stats: NetworkStats) => void;
  onUserPublished?: (user: IAgoraRTCRemoteUser, mediaType: 'video' | 'au
  onUnpublished?: (user: IAgoraRTCRemoteUser, mediaType: 'video' | '
}
```

Complete Example

Here's a comprehensive example demonstrating how to use the SDK:

```
import { GenericAgoraSDK } from 'akool-streaming-avatar-sdk';

// Initialize the SDK
const agoraSDK = new GenericAgoraSDK({ mode: "rtc", codec: "vp8" });

// Set up event handlers
agoraSDK.on({
  onStreamMessage: (uid, message) => {
    console.log("Received message from", uid, ":", message);
  },
  onException: (error) => {
    console.error("An exception occurred:", error);
  },
  onMessageReceived: (message) => {
    console.log("New message:", message);
    // Update UI with new message
    updateMessageDisplay(message);
  },
  onMessageUpdated: (message) => {
    console.log("Message updated:", message);
    // Update existing message in UI
    updateExistingMessage(message);
  },
  onNetworkStatsUpdated: (stats) => {
    console.log("Network stats:", stats);
    // Update network quality indicator
    updateNetworkQuality(stats);
  },
  onTokenWillExpire: () => {
    console.log("Token will expire in 30s");
    // Refresh token from backend
    refreshToken();
  },
  onTokenDidExpire: () => {
    console.log("Token expired");
    // Handle token expiration
    handleTokenExpiry();
  },
  onUserPublished: async (user, mediaType) => {
    if (mediaType === 'video') {
      const remoteTrack = await agoraSDK.getClient().subscribe(user, mec
      remoteTrack?.play('remote-video-container');
    } else if (mediaType === 'audio') {
      const remoteTrack = await agoraSDK.getClient().subscribe(user, mec
      remoteTrack?.play();
    }
  }
});

// Function to initialize session
async function initializeSession() {
  try {
    // Get session credentials from your backend
    const response = await fetch('/api/get-agora-credentials');
    const credentials = await response.json();

    // Join the Agora channel
    await agoraSDK.joinChannel({
      agora_app_id: credentials.agora_app_id,
      agora_channel: credentials.agora_channel,
      agora_token: credentials.agora_token,
      agora_uid: credentials.agora_uid
    });

    // Initialize avatar chat
    await agoraSDK.joinChat({
      vid: "your-voice-id",
      lang: "en",
      mode: 2
    });
    console.log("Session initialized successfully");
  } catch (error) {
    console.error("Failed to initialize session:", error);
  }
}

// Function to send a message
async function sendMessage(content) {
  try {
    await agoraSDK.sendMessage(content);
    console.log("Message sent successfully");
  } catch (error) {
    console.error("Failed to send message:", error);
  }
}

// Function to toggle microphone
async function toggleMicrophone() {
  try {
    await agoraSDK.toggleMic();
    const isEnabled = agoraSDK.isMicEnabled();
    console.log("Microphone is now:", isEnabled ? "enabled" : "disabled"
    updateMicButton(isEnabled);
  } catch (error) {
    console.error("Failed to toggle microphone:", error);
  }
}

// Function to clean up when leaving
async function cleanup() {
  try {
    await agoraSDK.closeStreaming();
    console.log("Session ended successfully");
  } catch (error) {
    console.error("Error during cleanup:", error);
  }
```

```
}

// Initialize the session
initializeSession();
```

Error Handling

The SDK provides comprehensive error handling through the `onException` event:

```
agoraSDK.on({
  onException: (error) => {
    console.error("SDK Error:", error.code, error.msg);

    // Handle specific error codes
    switch (error.code) {
      case 1001:
        // Handle authentication error
        handleAuthError();
        break;
      case 1002:
        // Handle network error
        handleNetworkError();
        break;
      default:
        // Handle other errors
        handleGenericError(error);
    }
  }
});
```

Requirements

- Node.js 14 or higher (for development)
- Modern browser with WebRTC support
- Valid Agora credentials
- Akool API access

Browser Support

- Chrome 56+
- Firefox 44+
- Safari 11+
- Edge 79+
- Opera 43+

Additional Resources

- Getting Started Guide
- Best Practices
- NPM Package
- GitHub Repository
- Akool API Documentation

‹ Streaming Avatar SDK Best Practices

Knowledge Base ›