AKOOL

Streaming Avatar › **Streaming Avatar SDK Best Practices**

Streaming Avatar

Streaming Avatar SDK Best Practices

Learn how to implement the Akool Streaming Avatar SDK securely and efficiently

Overview

This guide provides comprehensive best practices for implementing the **akool-streaming-avatar-sdk** package securely and efficiently. When implementing a JavaScript SDK that interacts with sensitive resources or APIs, it is critical to ensure the security of private keys, tokens, and other sensitive credentials.

Key Security Principles:

- Never expose private keys in client-side code
- Use short-lived session tokens
- Delegate authentication to a backend server
- Implement proper error handling and logging
- Follow secure coding practices for real-time communication

Package Information:

- NPM Package
- GitHub Repository

Current Version: 1.0.6

AKOOL

Prerequisites

Get your **Akool API Token** from **Akool Authentication API**

Basic knowledge of backend services and internet security

Understanding of JavaScript and HTTP requests

Node.js 14+ (for development)

Modern browser with WebRTC support

Security Best Practices

1. Backend Session Management

❌ Never do this (Client-side token exposure):

```
// BAD: Exposing sensitive credentials on client
const agoraSDK = new GenericAgoraSDK();
await agoraSDK.joinChannel({
  agora_app_id: "YOUR_SENSITIVE_APP_ID", // ❌ Exposed
  agora_token: "YOUR_SENSITIVE_TOKEN",   // ❌ Exposed
  agora_uid: 12345
});
```

✅ Do this instead (Secure backend delegation):

```
// GOOD: Get credentials from secure backend
const credentials = await fetch('/api/secure/get-agora-session', {
  method: 'POST',
  headers: { 'Authorization': `Bearer ${userToken}` },
  body: JSON.stringify({ userId, sessionType: 'avatar' })
});
const sessionData = await credentials.json();

await agoraSDK.joinChannel(sessionData.credentials);
```

2. Backend Implementation Example

Create secure endpoints in your backend to handle sensitive operations:

```
// Backend: Node.js/Express example
app.post('/api/secure/get-agora-session', authenticateUser, async (req,
  try {
    // Step 1: Get Akool access token
    const akoolToken = await getAkoolAccessToken();

    // Step 2: Create avatar session
    const sessionData = await createAvatarSession(akoolToken, req.body);

    // Step 3: Return only necessary data to client
    res.json({
      credentials: sessionData.credentials,
      sessionId: sessionData.id,
      expiresIn: sessionData.expires_in
    });
  } catch (error) {
    res.status(500).json({ error: 'Session creation failed' });
  }
});

async function getAkoolAccessToken() {
  const response = await fetch('https://openapi.akool.com/api/open/v3/ge
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      clientId: process.env.AKOOL_CLIENT_ID,      // Server-side env var
      clientSecret: process.env.AKOOL_SECRET_KEY  // Server-side env var
    })
  });

  const { data } = await response.json();
  return data.token;
}

async function createAvatarSession(token, sessionConfig) {
```

```
  const response = await fetch('https://openapi.akool.com/api/open/v4/li
    method: 'POST',
    headers: {
      'x-api-key': '{{API Key}}'
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      avatar_id: sessionConfig.avatarId || "dvp_Tristan_cloth2_1080P",
      duration: sessionConfig.duration || 300
    })
  });

  return await response.json();
}
```

4. Sign up functions for steaming events and button click events:

```
function handleStreamReady(event: any) {
  console.log("Stream is ready:", event.detail);
}
function handleMessageReceive(event: any) {
  console.log("Message:", event.detail);
}
function handleWillExpire(event: any) {
  console.log("Warning:", event.detail.msg);
}
function handleExpired(event: any) {
  console.log("Warning:", event.detail.msg);
}
function handleERROR(event: any) {
  console.error("ERROR has occurred:", event.detail.msg);
}
function handleStreamClose(event: any) {
  console.log("Stream is close:", event.detail);
  // when you leave the page you'd better off the eventhandler
  stream.off(StreamEvents.READY, handleStreamReady);
  stream.off(StreamEvents.ONMESSAGE, handleMessageReceive);
  stream.off(StreamEvents.WILLEXPIRE, handleWillExpire);
  stream.off(StreamEvents.EXPIRED, handleExpired);
  stream.off(StreamEvents.ERROR, handleERROR);
  stream.off(StreamEvents.CLOSED, handleStreamClose);
}

stream.on(StreamEvents.READY, handleStreamReady);
stream.on(StreamEvents.ONMESSAGE, handleMessageReceive);
stream.on(StreamEvents.WILLEXPIRE, handleWillExpire);
stream.on(StreamEvents.EXPIRED, handleExpired);
stream.on(StreamEvents.ERROR, handleERROR);
stream.on(StreamEvents.CLOSED, handleStreamClose);

async function handleToggleSession() {
  if (
```

```
    window.toggleSession.innerHTML == "   ...  
    return;
  if (window.toggleSession.innerHTML == "Start Session") {
    window.toggleSession.innerHTML = "   ...  &
    await stream.startSessionWithCredentials(
      "yourStreamingVideoDom",
      paramsWithCredentials
    );
    window.toggleSession.innerHTML = "End Session";
    window.userInput.disabled = false;
    window.sendButton.disabled = false;
    window.voiceButton.disabled = false;
  } else {
    // info: close your stream session
    stream.closeStreaming();
    window.messageWrap.innerHTML = "";
    window.toggleSession.innerHTML = "Start Session";
    window.userInput.disabled = true;
    window.sendButton.disabled = true;
    window.voiceButton.disabled = true;
  }
}

async function handleSendMessage() {
  await stream.sendMessage(window.userInput.value ?? "");
}

async function handleToggleMic() {
  await stream.toggleMic();
  if (stream.micStatus) {
    window.voiceButton.innerHTML = "Turn mic off";
  } else {
    window.voiceButton.innerHTML = "Turn mic on";
  }
}
```

```
window.toggleSession.addEventListener("click", handleToggleSession);
window.sendButton.addEventListener("click", handleSendMessage);
window.voiceButton.addEventListener("click", handleToggleMic);
```

Additional Resources

- **Complete API Reference** - Detailed API documentation
- **Getting Started Guide** - Quick start tutorial
- **NPM Package** - Official package
- **GitHub Repository** - Source code
- **Akool Authentication API** - Authentication guide
- **Error Codes Reference** - Error handling guide
- **Live Avatar API** - Backend session creation

‹ Streaming Avatar SDK Quick Start

Streaming Avatar SDK API Reference ›