

Cloud Contact Center SDK Development Guide Product Documentation



Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

SDK Development Guide

Getting Started

Integrating Agent SDK

Using Demo for Quick Operation

Web

uni-app

Android

iOS

Initializing SDK

Web

uni-app

Android

iOS

Agent SDK APIs

Web

uni-app

Android

iOS

常见问题

FAQs About Web SDK

FAQs About uni-app SDK

FAQs About Client SDK

Integrated Telephony Customer Service

Implementing One-Click Outbound Call

Web

uni-app

Android

IOS

Implementing Call-In Function

Web

SDK Development Guide

Getting Started

Last updated : 2024-04-01 17:18:07

Tencent Cloud Contact Center (Cloud Contact Center) helps enterprises quickly set up a customer contact platform that integrates phone, online communication, and audio and video calls. Cloud Contact Center provides a full set of SDKs for customer contact. By reading this topic, you can understand the process of implementing customer contact according to the SDK.

Getting Started Guide

You can follow these steps for integrated development:

Step	Operation
1	Creating Cloud Contact Center application
2	Referring to the required customer service type, configure accordingly: Quick configuration for outbound calls Quick configuration for inbound calls
3	Refer to Integrating Agent-side SDK to integrate the agent side into your own system
4	Refer to the corresponding documents for the customer service type you need to integrate: (It's more convenient to navigate from the sidebar) Integrating Telephone Customer Service

Exchange and Feedback

[Click here to enter the Cloud Contact Center community](#), where you can get support from professional engineers to solve your problems.

Integrating Agent SDK Using Demo for Quick Operation Web

Last updated : 2024-04-01 17:21:52

We have provided demos under different frameworks, which you can download and run quickly:

[Vue Demo](#)

[React Demo](#)

After the demo is downloaded, run it according to the guidance of `README.md`. You may also integrate it into your own project based on the subsequent documentation.

Exchange and Feedback

[Click here to enter the Cloud Contact Center community](#), where you can get support from professional engineers to solve your problems.

uni-app

Last updated : 2024-04-01 17:30:34

This topic mainly introduces how to quickly run the Cloud Contact Center uni-app Demo.

Environment Requirements

We recommend using the latest HBuilderX editor.

An iOS device running iOS 9.0 or later and supporting audio.

An Android device running on a version not earlier than 4.1 and supporting audio. Simulators are not currently supported. The option that allows debugging must be enabled.

Your iOS/Android device has been connected to the internet.

Prerequisites

You have [signed up for a Tencent Cloud](#) account.

You have [activated Cloud Contact Center](#) service and created a [Cloud Contact Center instance](#).

You have completed [Connecting Your Own Number](#). You have also finished the corresponding [IVR configuration](#).

Key Concepts

SdkAppId: The application ID users create on the [Cloud Contact Center console](#). You can create up to 20 Cloud Contact Center applications under one Tencent Cloud account, often starting with 140.

UserID: The account configured by the agent or administrator in the Cloud Contact Center, usually in the format of an email address. After the application is created for the first time, the main account can go to [Internal Message](#) (sub-account requires a subscription to Cloud Contact Center product messages) to view the contact center administrator account and password. Under one SDKAppID, multiple UserIDs can be configured. If the configuration limit is exceeded, more agents need to be purchased in [Agent Purchase](#).

SecretId and SecretKey: A certificate needed by developers to call cloud APIs, created on the [Tencent Cloud console](#).

Token: A sign-in ticket, which requires calling the Application Programming Interface [CreateSDKLoginToken](#) to access. The correct approach is to place the token calculation code and encryption key on your business server, then the app requests access to the token calculated in real time from your server as needed.

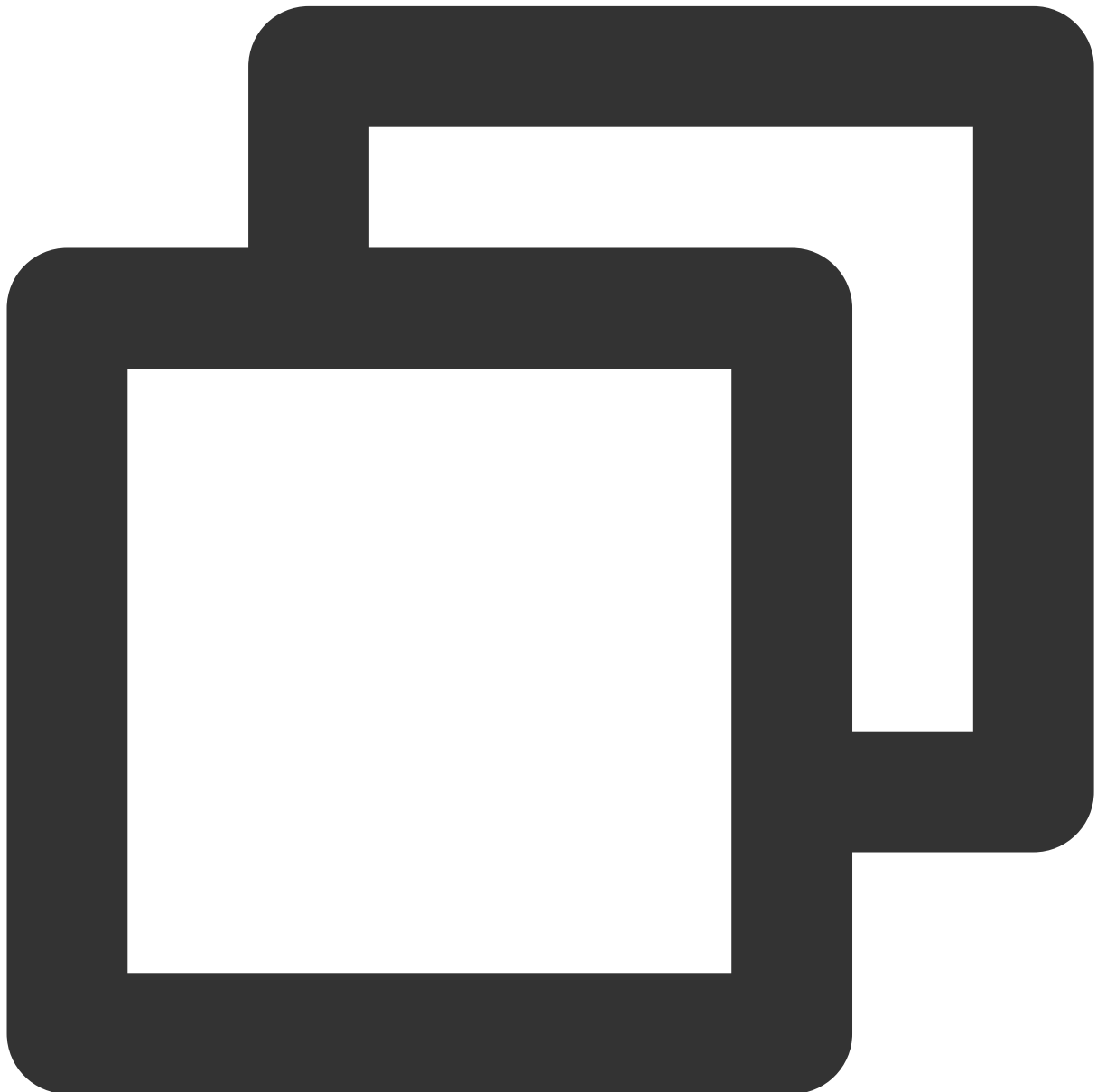
Operation Steps

Step 1: Download the tccc-agent-uniapp-example Source Code

Download the [tccc-agent-uniapp-example](#) source code based on actual business needs.

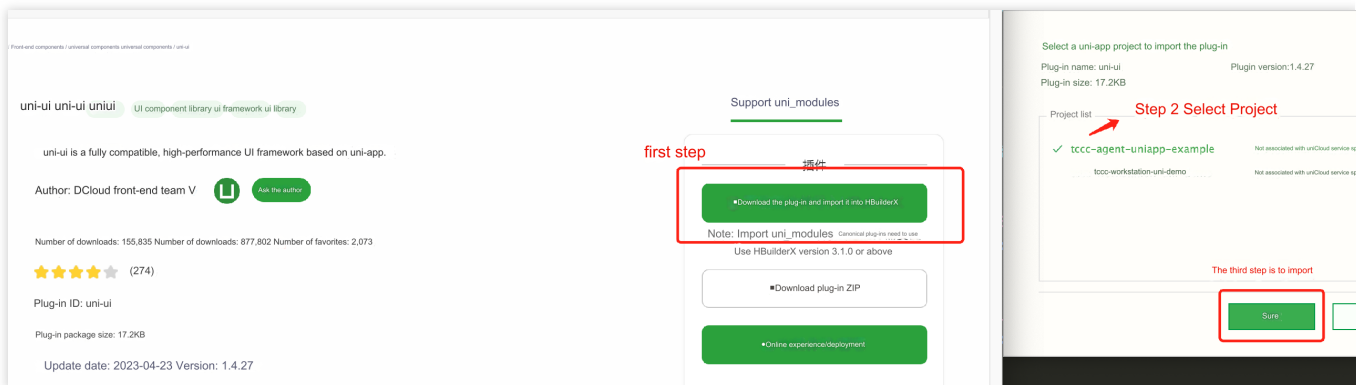
Step 2: Install Dependencies

Install npm package dependency.



```
npm i tccc-sdk-uniapp
```

Install uni-ui. Use HBuilderX to import **uni-ui**.



Step 3: Configure the tccc-agent-uniapp-example Project File

1. Find and open the debug/genTestToken.js file.
2. Set related parameters in the genTestToken.js file:

USERID: Agent account, format: `xxx@qq.com`

SDKAPPID: Cloud Contact Center SDKAppId, which needs to be replaced with your own account's SDKAppId

SECRETID: The ID of the encryption key used to calculate the signature

SECRETKEY: The key of the encryption key used to calculate the signature

```
genTestToken.js
1 day/**
2  * Agent account, the format is: xxx@qq.com
3  */
4  const USERID = 'xxx@qq.com';
5
6th/**
7  * Tencent Cloud SDKAppId needs to be replaced with the SDKAppId under your own account.
8  * Enter Tencent Cloud Call Center [Console] (https://console.cloud.tencent.com)
9  * It is the unique identifier used by Tencent Cloud to distinguish customers.
10 */
11 const SDKAPPID = 1400000000;
12
13 7** on the 12th
14  * Encryption key ID used to calculate signature, [View secret key](https://console.cloud.tencent.com/secretkey)
15  * Note: This solution is only suitable for debugging Demo. Please change the UserSig calculation code and password before officially going online.
16  * Document: https://cloud.tencent.com/document/product/679/58260
17  const SECRETID = '';
18
19th/**
20  * Encryption key Key used to calculate signature, [View key](https://console.cloud.tencent.com/secretkey)
21  * Note: This solution is only suitable for debugging Demo. Please change the UserSig calculation code and password before officially going online.
22  * Document: https://cloud.tencent.com/document/product/679/58260
23  */
24  const SECRETKEY = '';
25
```

Note:

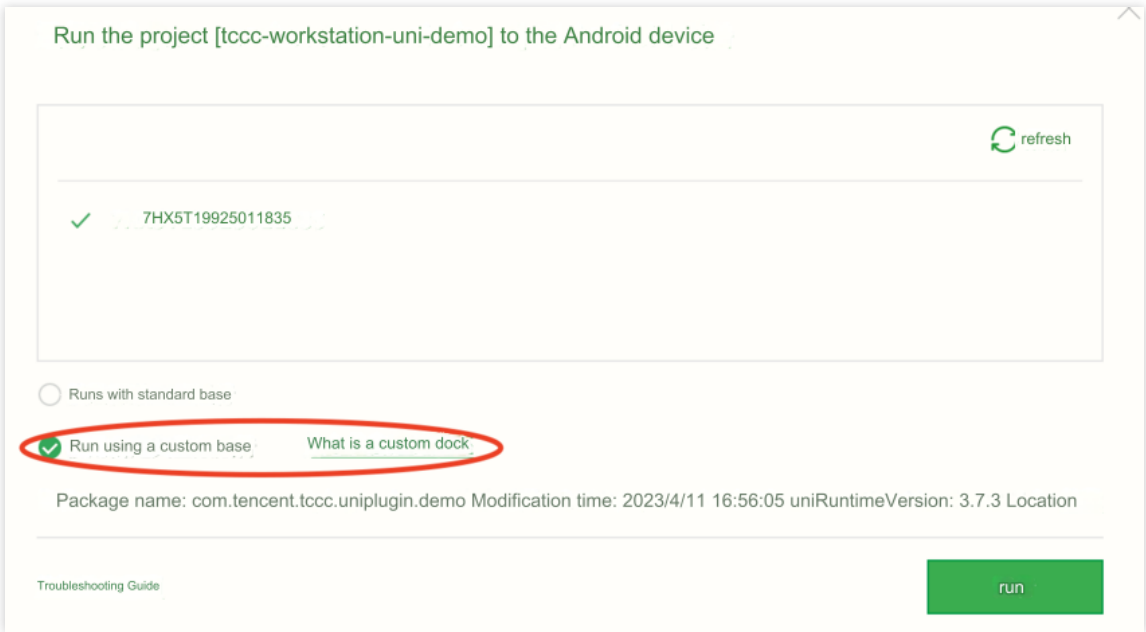
Please do not publish the following code in the online version of your app. The reasons are as follows:

Although the code in this file can correctly calculate the token, it is only suitable for quickly implementing the basic features of the SDK, **but not suitable for online products**. This is because the SECRETKEY in the client code is easily decompiled and reverse-engineered, especially in the case of web-based code where the difficulty of cracking is almost negligible. Once your key is leaked, attackers can calculate the correct token to steal your Tencent Cloud traffic.

The correct approach is to place the token calculation code and encryption key on your business server, and the app requests a token calculated in real time from your server when necessary. The cost of cracking the server is higher than the cost of cracking the client app, so the server calculation scheme can better protect your encryption key. For details, refer to [Creating SDK Login Token?](#).

Step 4: Compile the Demo

Use **Self-Defined Stand Packaging Run** (do not choose standard stand run) , and use **physical machine run** for the self-defined stand.



Note:

For details on a self-defined debugging stand and how to use it, please refer to [the official tutorial](#).

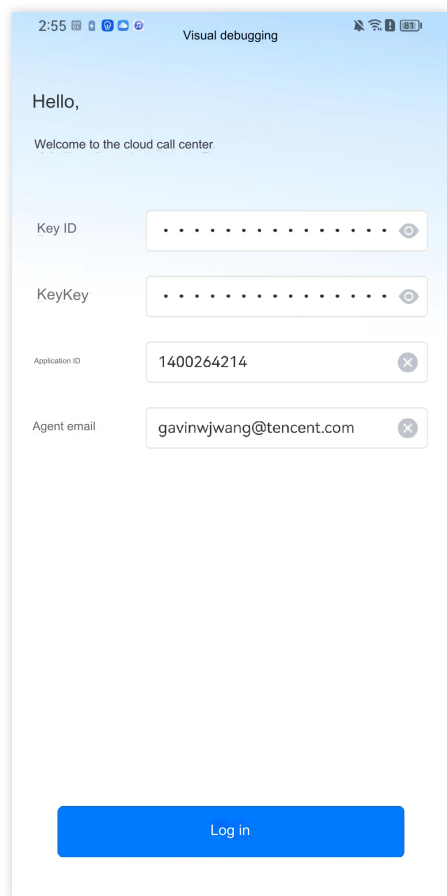
Step 5: Run the Demo

- 1. After choosing to run on the **physical machine**, click **Log In**.
- 2. After successfully logging in, enter the phone number you need to dial to use the dial feature.

Execution Effect

The basic features are shown in the figure below:

Login Page	Number Management Page	Dial Pa



2:55 Visual debugging

Hello,

Welcome to the cloud call center

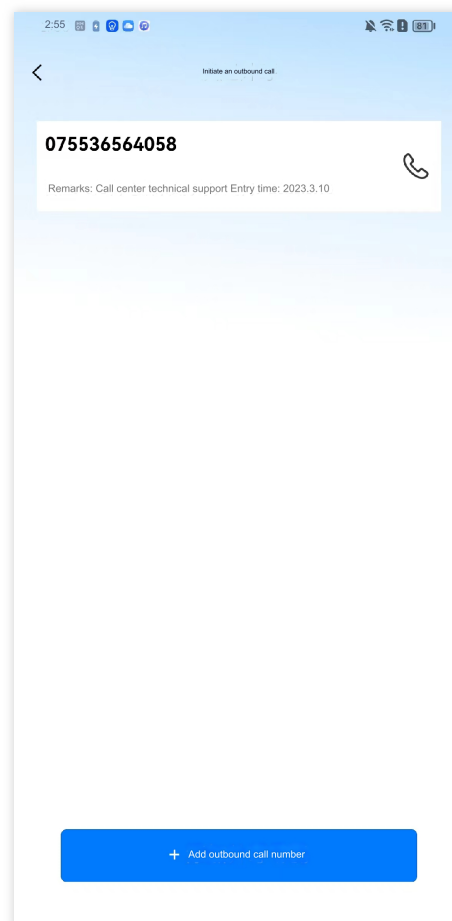
Key ID

KeyKey

Application ID

Agent email

Log in



2:55 Initiate an outbound call

075536564058

Remarks: Call center technical support Entry time: 2023.3.10

+ Add outbound call number

Exchange and Feedback

[Click here to enter the Cloud Contact Center community](#), where you can get support from professional engineers to solve your problems.

Android

Last updated : 2024-04-01 17:32:40

Quickly Running Cloud Contact Center Android Demo

Cloud Contact Center provides an Android SDK, which allows agents to make calls through landline phones. It can also implement outbound calls on mobile phones and PCs, and inbound call reception through our SDK.

This topic mainly introduces how to quickly run the Cloud Contact Center Android Demo. Just follow the steps for configuration, and you can run the related features of Cloud Contact Center.

Environment Requirements

Android Studio 3.5+.

Android 4.1 (SDK API 16) or later.

Prerequisites

You have [signed up for a Tencent Cloud](#) account

You have [activated Cloud Contact Center](#) service and created a [Cloud Contact Center instance](#).

You have completed [Connecting Your Own Number](#). You have also finished the corresponding [IVR configuration](#).

Key Concepts

SdkAppId: The application ID users create on the [Cloud Contact Center console](#). You can create up to 20 Cloud Contact Center applications under one Tencent Cloud account, often starting with 140.

UserID: The account configured by the agent or administrator in the Cloud Contact Center, usually in the format of an email address. After the application is created for the first time, the main account can go to [Internal Message](#) (sub-account requires a subscription to Cloud Contact Center product messages) to view the contact center administrator account and password. Under one SDKAppID, multiple UserIDs can be configured. If the configuration limit is exceeded, more agents need to be purchased in [Agent Purchase](#).

SecretId and SecretKey: A certificate needed by developers to call cloud APIs, created on the [Tencent Cloud console](#).

Token: Sign-in ticket, required to call the Application Programming Interface [CreateSDKLoginToken](#) to access. The correct approach is to place the token calculation code and encryption Key on your business server, and then the App

requests access to the token calculated in real time from your server when necessary.

Operation Steps

Step 1: Download the tccc-agent-java-example Source Code

Download the [tccc-agent-java-example](#) source code based on actual business needs.

Step 2: Configure the tccc-agent-java-example Project File

1. Find and open the

`debug/src/main/java/com/tencent/tccsdk/debug/GenerateTestUserToken.java` file.

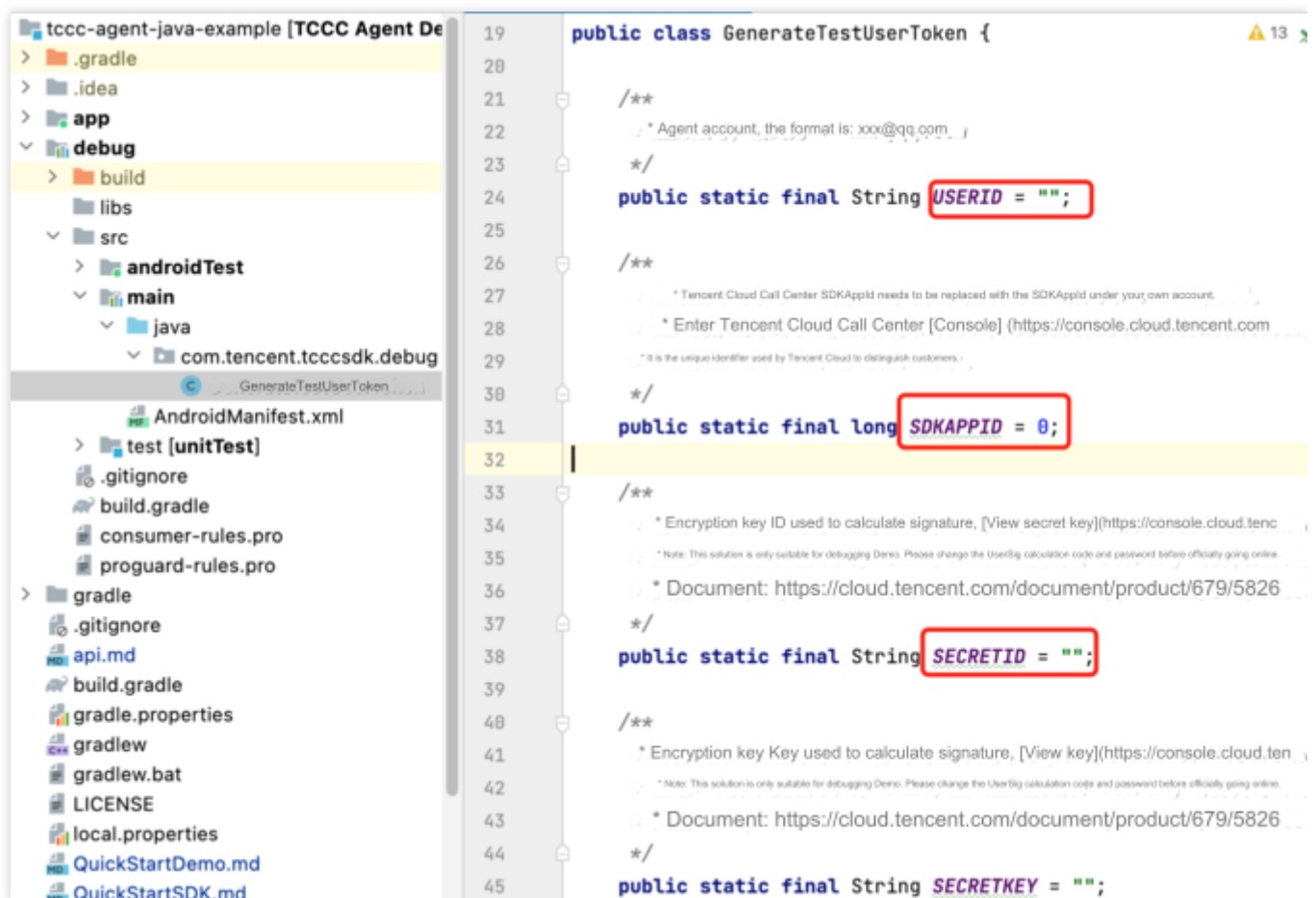
2. Set related parameters in the `GenerateTestUserToken.java` file:

USERID: Agent account, format: xxx@qq.com

SDKAPPID: Cloud Contact Center SDKAppId, which needs to be replaced with your own account's SDKAppId

SECRETID: The ID of the encryption key used to calculate the signature

SECRETKEY: The key of the encryption key used to calculate the signature



Note:

Please do not publish the following code in the online version of your app. The reasons are as follows:

Although the code in this file can correctly calculate the token, it is only suitable for quickly implementing the basic features of the SDK, **but not suitable for online products**. This is because the SECRETKEY in the client code is easily decompiled and reverse-engineered, especially in the case of web-based code where the difficulty of cracking is almost negligible. Once your key is leaked, attackers can calculate the correct token to steal your Tencent Cloud traffic.

The correct approach is to place the token calculation code and encryption key on your business server, and the app requests a token calculated in real time from your server when necessary. The cost of cracking the server is higher than the cost of cracking the client app, so the server calculation scheme can better protect your encryption key. For details, refer to [Creating SDK Login Token?](#).

Step 3: Compile and Run the Demo

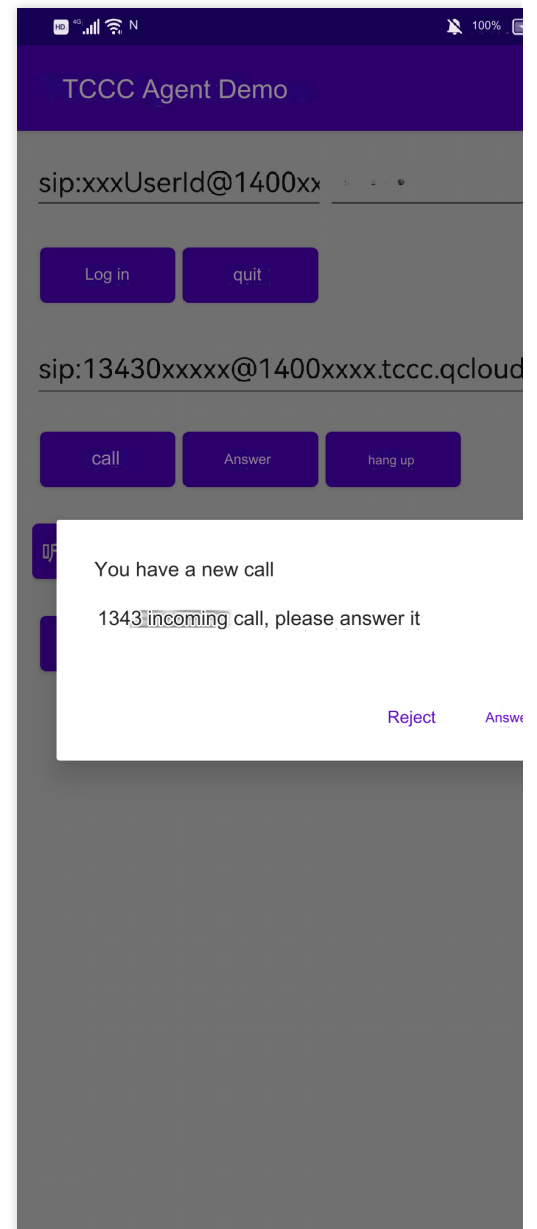
Open source code project `tccc-agent-java-example` with Android Studio (version 3.5 or later), and click **Run**.

1. Click **Log In**.
2. After successfully logging in, enter the phone number you need to dial to use the dial feature.

Execution Effect

The basic features are shown in the figure below:

Call Effect	Answer Effect



Exchange and Feedback

[Click here to enter the Cloud Contact Center community](#), where you can get support from professional engineers to solve your problems.

iOS

Last updated : 2024-04-01 17:34:41

Quickly Running Cloud Contact Center iOS Demo

Cloud Contact Center provides an iOS SDK, which allows agents to make phone calls and use other features. It can also implement outbound calls on mobile phones and PCs, and inbound call reception through our SDK.

This topic mainly introduces how to quickly run the Cloud Contact Center iOS Demo. Just follow the steps for configuration, and you can run the related features of Cloud Contact Center.

Environment Requirements

Xcode 9.0+.

A real iPhone or iPad running iOS 9.0 or later.

The project has been configured with a valid developer signature.

Prerequisites

You have [signed up for Tencent Cloud](#).

You have [activated Cloud Contact Center](#) service and created a [Cloud Contact Center instance](#).

You have completed [Connecting Your Own Number](#). You have also finished the corresponding [IVR configuration](#).

Key Concepts

1. **SdkAppId**: The application ID users create on the [Cloud Contact Center console](#). You can create up to 20 Cloud Contact Center applications under one Tencent Cloud account, often starting with 140.
2. **UserID**: The account configured by the agent or administrator in the Cloud Contact Center, usually in the format of an email address. After the application is created for the first time, the main account can go to [Internal Message](#) (sub-account requires a subscription to Cloud Contact Center product messages) to view the contact center administrator account and password. Under one SDKAppID, multiple UserIDs can be configured. If the configuration limit is exceeded, more agents need to be purchased in [Agent Purchase](#).
3. **SecretId and SecretKey**: A certificate needed by developers to call cloud APIs, created on the [Tencent Cloud console](#).

4. **Token:** A sign in ticket, which requires calling the cloud API interface [CreateSDKLoginToken](#) to access. The correct approach is to place the token calculation code and encryption Key on your business server, and then have the app request access to a token calculated in real time from your server.

Operation Steps

Step 1: Download the tccc-agent-ios-example Source Code

Download the [tccc-agent-ios-example](#) source code based on actual business needs.

Step 2: Configure the tccc-agent-ios-example Project File

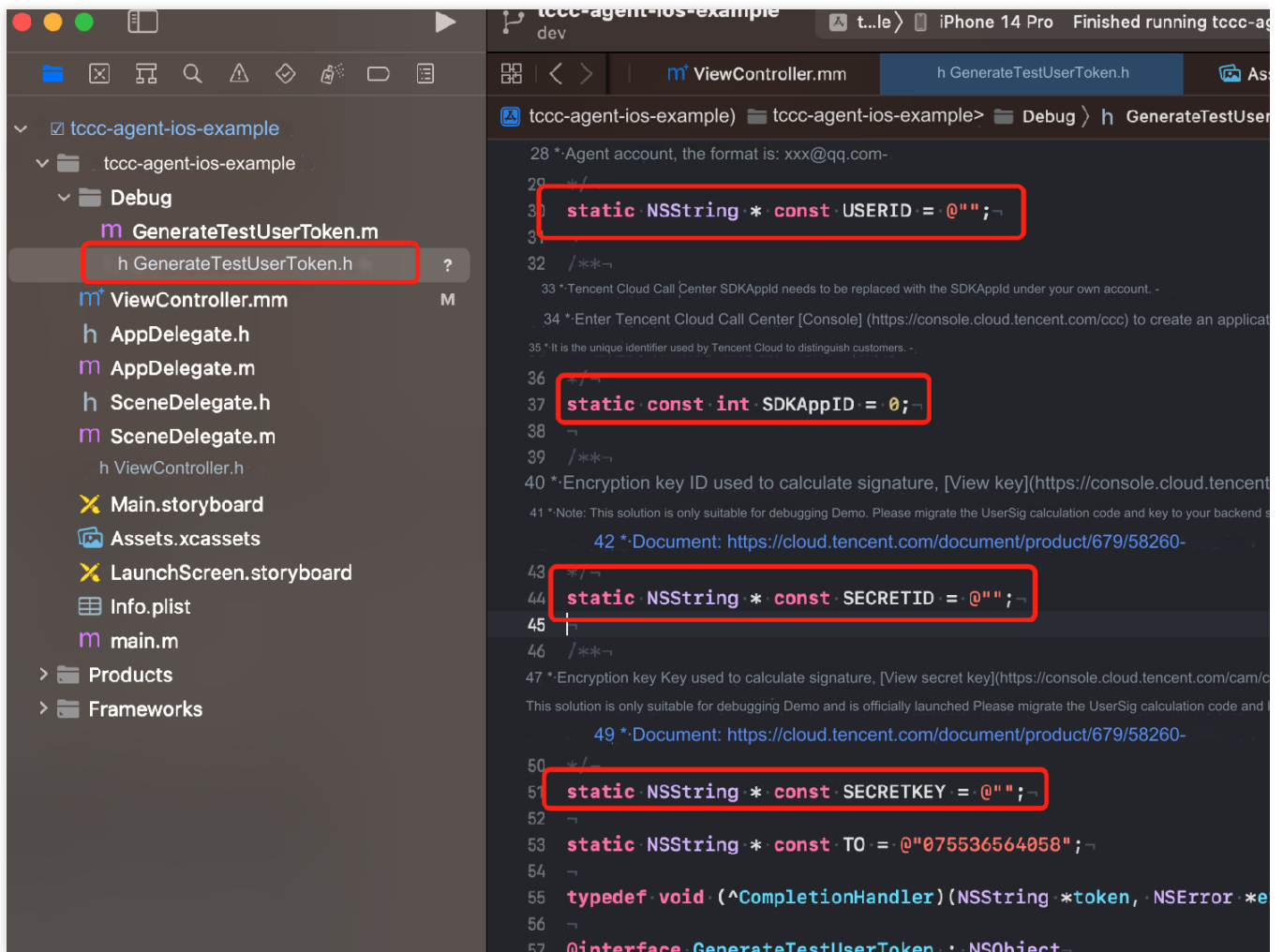
1. Find and open the debug/GenerateTestUserToken.h file.
2. Set related parameters in the GenerateTestUserToken.h file:

USERID: Agent account, format : xxx@qq.com

SDKAppID: Cloud Contact Center SDKAppId, which needs to be replaced with your own account's SDKAppId

SECRETID: The ID of the encryption key used to calculate the signature

SECRETKEY: The key of the encryption key used to calculate the signature



Warning:

Please do not publish the following code in the online version of your app. The reasons are as follows:

Although the code in this file can correctly calculate the token, it is only suitable for quickly implementing the basic features of the SDK, but not suitable for online products. This is because the SECRETKEY in the client code is easily decompiled and reverse-engineered, especially in the case of web-based code where the difficulty of cracking is almost negligible. Once your key is leaked, attackers can calculate the correct token to steal your Tencent Cloud traffic.

The correct approach is to place the token calculation code and encryption Key on your business server, and then have the app request access to a token calculated in real time from your server. Since the cost of cracking the server is higher than that of cracking the client app, the server calculation scheme better protects your encryption Key. For more details, please see [Creating SDK Sign in Token](#)

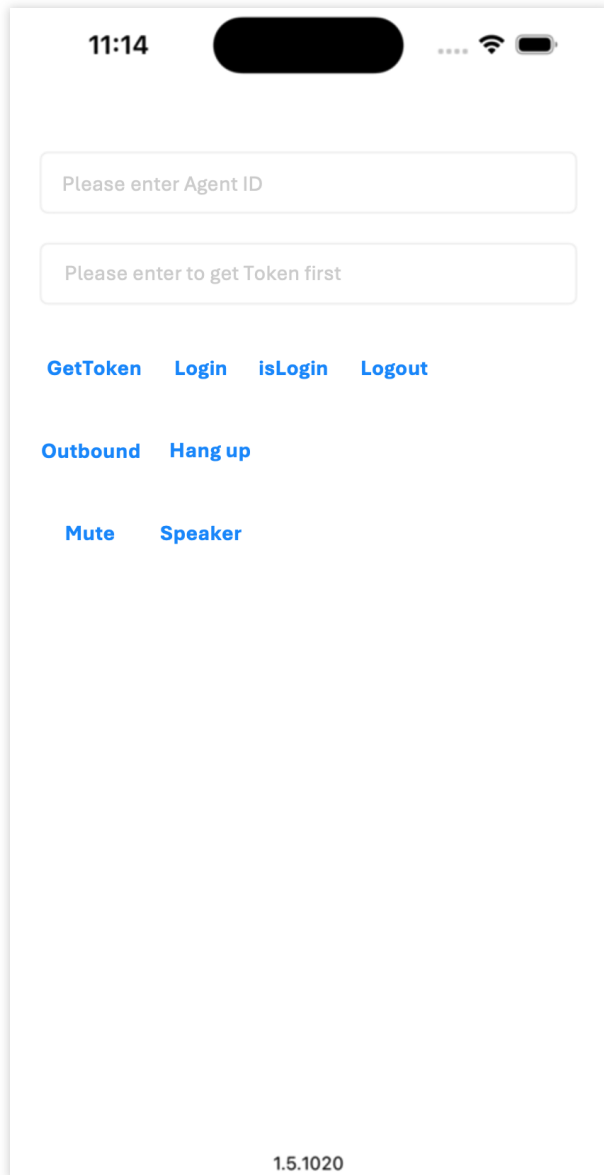
Step 3: Compile and Run the Demo

Open the source code project `tccc-agent-ios-example` with Xcode and click **Run**.

1. Click **Obtain Token > Log In**.
2. After successfully logging in, click **Outbound Call** to use the dialing feature.

Execution Effect

The basic features are shown in the figure below:



Exchange and Feedback

[Click here to enter the Cloud Contact Center community](#), where you can get support from professional engineers to solve your problems.

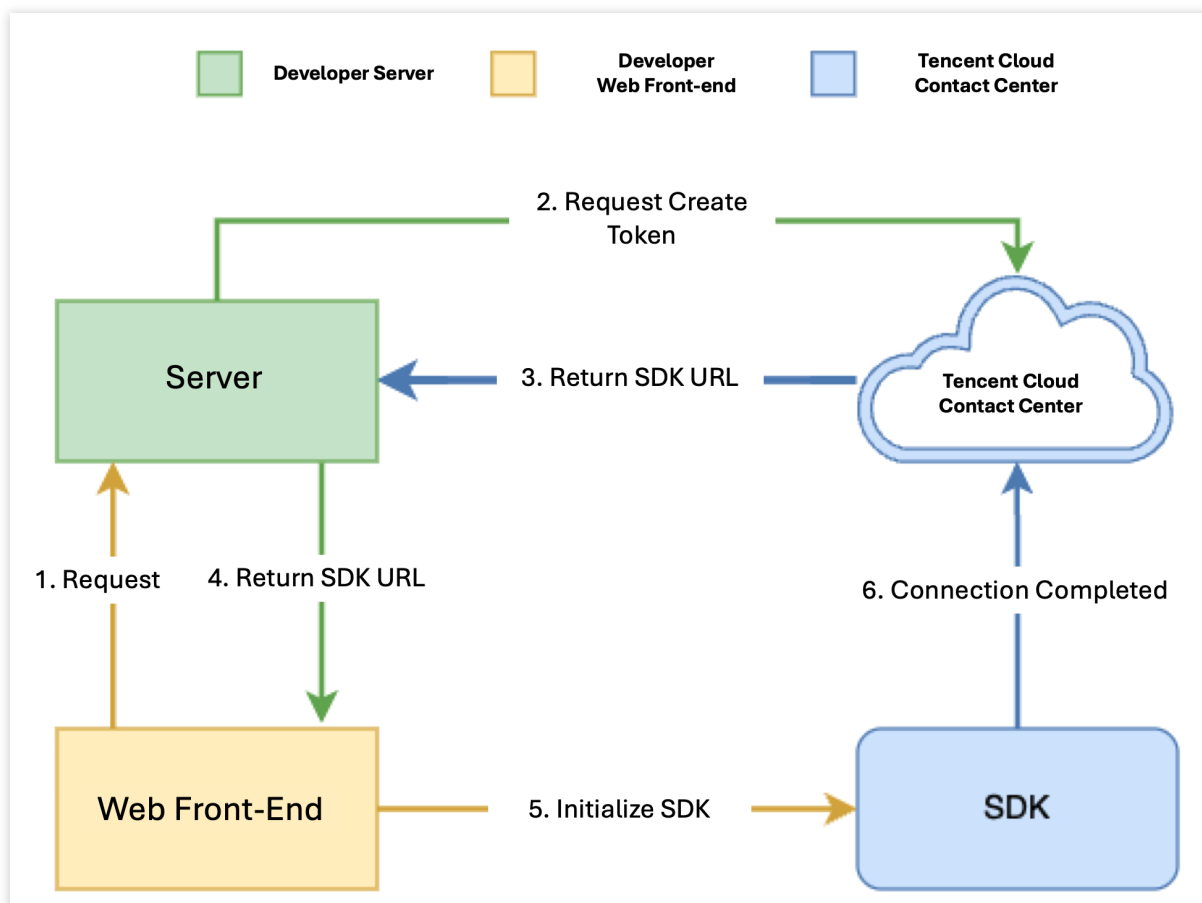
Initializing SDK

Web

Last updated : 2024-04-01 18:02:33

Principles

Cloud Contact Center provides a JavaScript SDK for developers. Developers can import the SDK into the page using a script, which completes the initialization of the SDK. The integration interaction is as follows:



Notes

1. Cloud Contact Center's agent-side Web SDK primarily supports Google Chrome 56 and later and Microsoft Edge 80 and later. It is recommended to install the latest version of the browser to support more features.

2. Please use the HTTPS protocol to deploy the front-end page (localhost can be used during development), otherwise, normal calls will not be possible due to browser restrictions.

Integration Prerequisites

1. You have created a [Cloud Contact Center application](#).
2. You have purchased and added an [agent account](#).

Key Concepts

SdkAppId: The application ID users create on the [Cloud Contact Center console](#). You can create up to 20 Cloud Contact Center applications under one Tencent Cloud account, often starting with 140.

UserID: The account configured by the agent or administrator in the Cloud Contact Center, usually in the format of an email address. After the application is created for the first time, the main account can go to [Internal Message](#) (sub-account requires a subscription to Cloud Contact Center product messages) to view the contact center administrator account and password. Under one SDKAppID, multiple UserIDs can be configured. If the configuration limit is exceeded, more agents need to be purchased in [Agent Purchase](#).

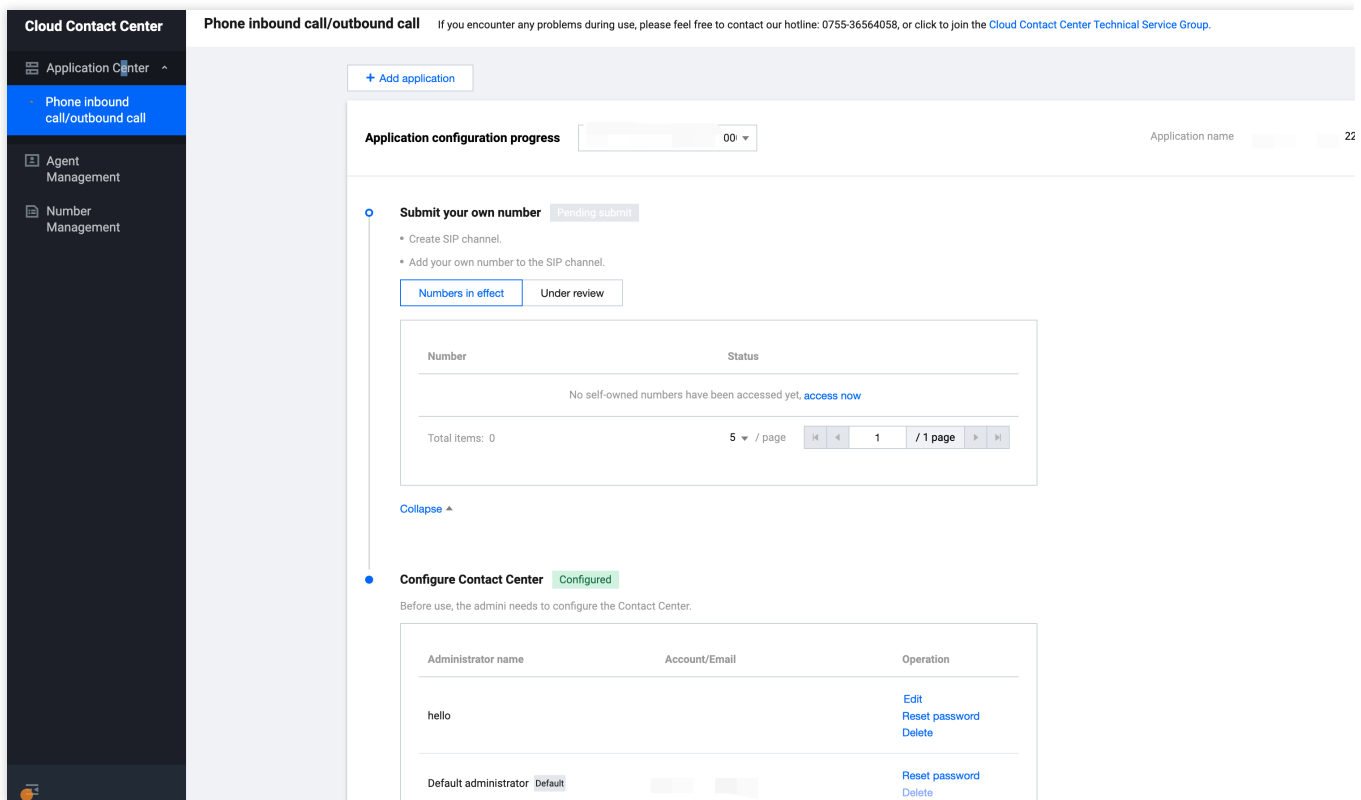
SecretId and SecretKey: A certificate needed by developers to call cloud APIs, created on the [Tencent Cloud console](#).

SDKURL: The JS URL needed to initialize the Web SDK, created via cloud APIs. The URL is valid for 10 minutes. Please ensure it is used only once and created when SDK needs to be initialized. No need to re-create it after successful SDK initialization.

SessionId: The unique ID for a user from the beginning to the end of a call. Developers can associate different recordings, service records, and event pushes via SessionId.

Step 1: Obtain Required Parameters

1. Obtain the `SecretId` and `SecretKey` of Tencent Cloud account. For details, refer to [Access Key](#).
2. Obtain the sdkAppId of the Cloud Contact Center application, and log in to the [Cloud Contact Center console](#) to view:

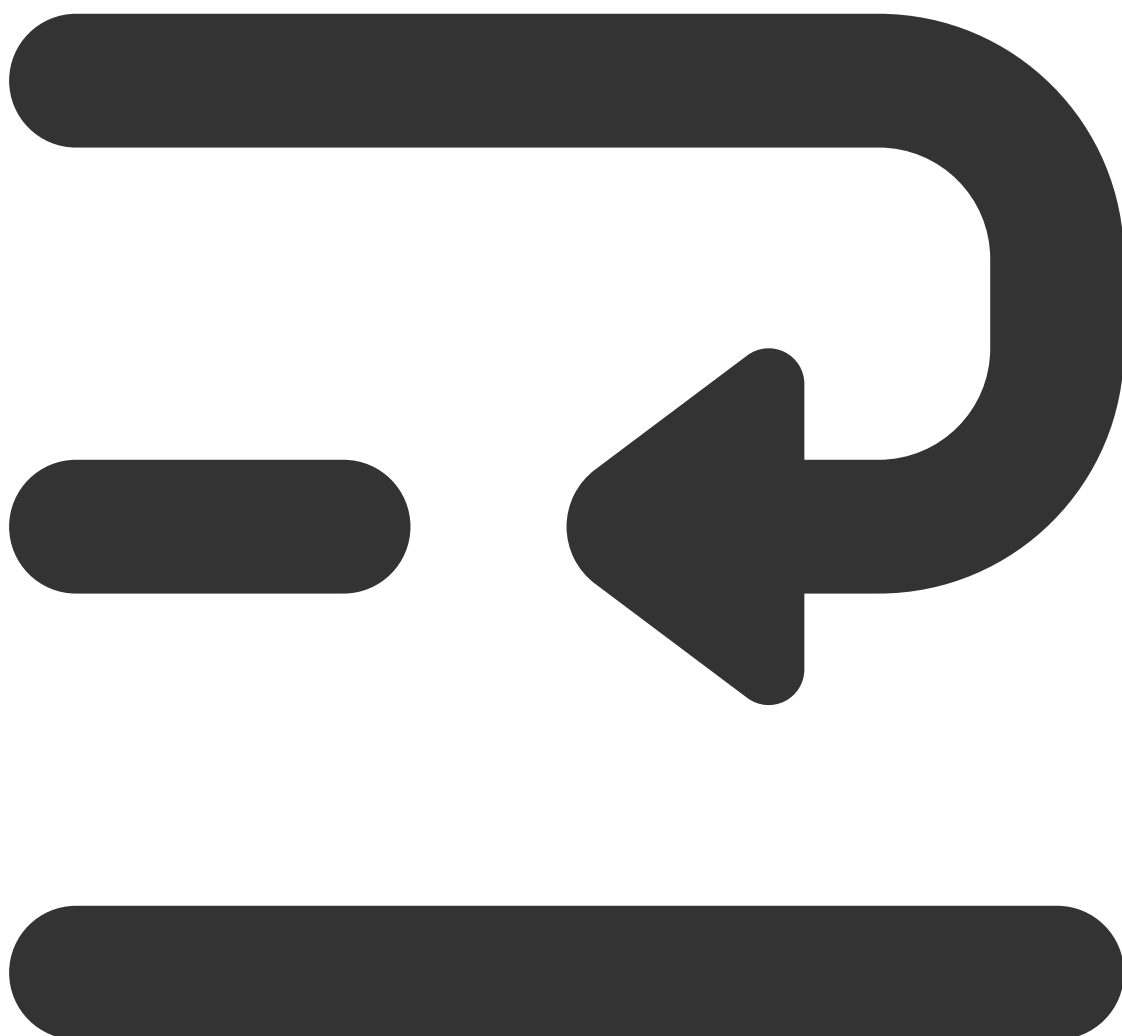


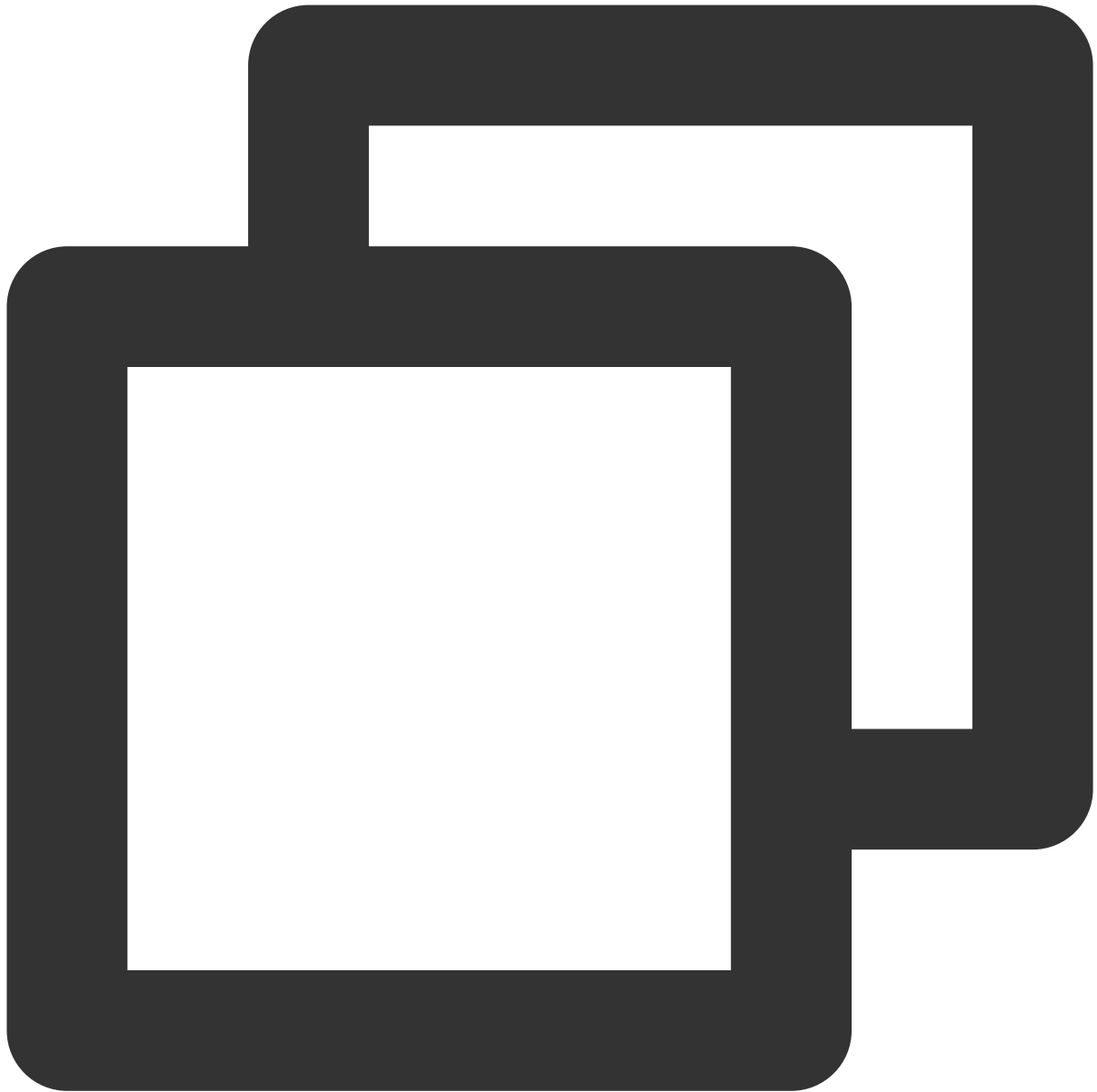
Step 2: Obtain the SDK URL

Note: This step requires backend development by the accessing party.

1. Integrate tencentcloud-sdk
2. Call the [CreateSDKLoginToken](#) interface.
3. Return the obtained SdkURL to the frontend of the accessing party.

In the following, we will use the interface name `/loginTCCC` to explain the interface developed in this step. The following code is an example in Node.js. For sample code in other languages, see [CreateSDKLoginToken](#).





```
// Version of tencentcloud-sdk-nodejs required to be 4.0.3 or later
const tencentcloud = require('tencentcloud-sdk-nodejs');
const express = require('express');
const app = express();
const CccClient = tencentcloud.ccc.v20200210.Client;

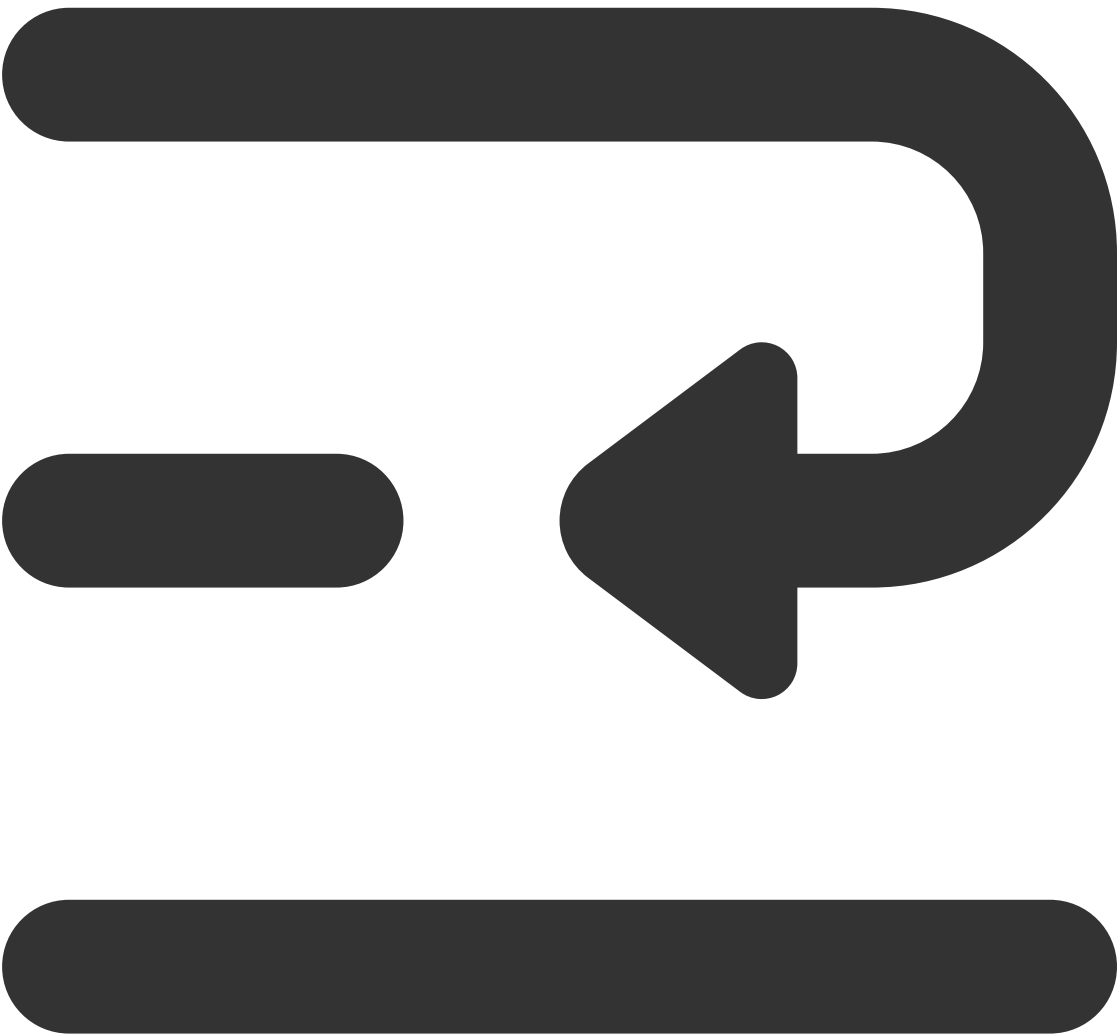
app.use('/loginTCCC', (req, res) => {
  const clientConfig = {
    // Address to obtain secretId and secretKey: https://console.tencentcloud.com/c
    credential: {
      secretId: 'SecretId',
```

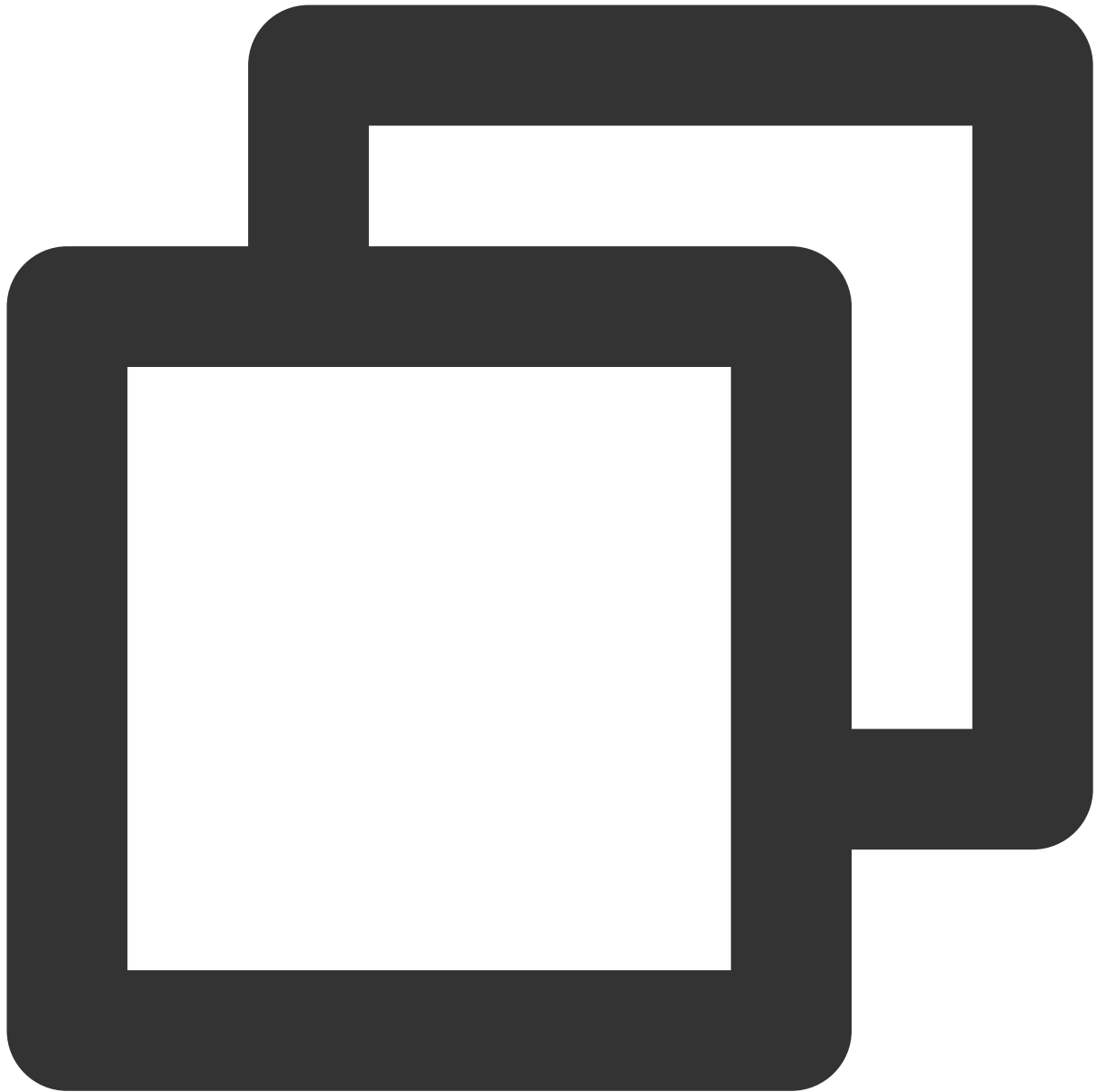
```
    secretKey: 'SecretKey'
  },
  region: '',
  profile: {
    httpProfile: {
      endpoint: 'ccc.tencentcloudapi.com'
    }
  }
};
const client = new CccClient(clientConfig);
const params = {
  SdkAppId: 1400000000, // Replace it with your own SdkAppId
  SeatUserId: 'xxx@qq.com' // Replace it with the agent account
};
client.CreateSDKLoginToken(params).then(
  (data) => {
    res.send({
      SdkURL: data.SdkURL
    })
  },
  (err) => {
    console.error('error', err);
    res.status(500);
  }
);
})
```

Step 3: Request to Obtain SDK URL and Complete Initialization from the Web Frontend

Note: This step requires frontend development by the accessing party.

1. Request the `/loginTCCC` interface implemented in step 2 and obtain the SdkURL.
2. Insert the SdkURL into the page using the script method.
3. After the `tccc.events.ready` event has been successfully listened to, execute the business logic.





```
function injectTcccWebSDK(SdkURL) {  
  if (window.tccc) {  
    console.warn('The SDK has been initialized. Please confirm whether the initia  
    return;  
  }  
  return new Promise((resolve, reject) => {  
    const script = document.createElement('script');  
    script.setAttribute('crossorigin', 'anonymous');  
    // The DomId needs to be rendered into it. If it is filled in, there is no fl  
    // To ensure the integrity of the workspace UI, the rendered Dom should have  
    // script.dataset.renderDomId = "renderDom";  
  })  
}
```

```
script.src = SdkURL;
document.body.appendChild(script);
script.addEventListener('load', () => {
  // JS SDK file loaded successfully. At this point, you can use the global v
  window.tccc.on(window.tccc.events.ready, () => {
    /**
     * TCCC SDK initialization succeeded. At this point, you can call out, lis
     * Note ⚠: Please ensure that the SDK is only initialized once
     * */
    resolve('Initialization succeeded')
  });
  window.tccc.on(window.tccc.events.tokenExpired, ({message}) => {
    console.error('Initialization failed', message)
    reject(message)
  })
})
})
}

// Request the interface implemented in step 2 /loginTCCC
// Note ⚠: The following is just a code example, it is not recommended to run direct
fetch('/loginTCCC')
  .then(res => res.json())
  .then((res) => {
    const SdkURL = res.SdkURL; // Ensure that the SdkURL is returned by the request
    return injectTcccWebSdk(SdkURL);
  })
  .catch((error) => {
    // Initialization failed
    console.error(error);
  })
})
```


uni-app

Last updated : 2024-04-01 17:43:18

This topic mainly introduces how to quickly integrate Cloud Contact Center uni-app SDK into your project.

Environment Requirements

We recommend using the latest HBuilderX editor.

An iOS device running iOS 9.0 or later and supporting audio.

An Android device running on a version not earlier than 4.1 and supporting audio. Simulators are not currently supported. The option that allows debugging must be enabled.

Your iOS/Android device has been connected to the internet.

Integration Prerequisites

You have [signed up for a Tencent Cloud](#) account

You have [activated Cloud Contact Center](#) service and created a [Cloud Contact Center instance](#).

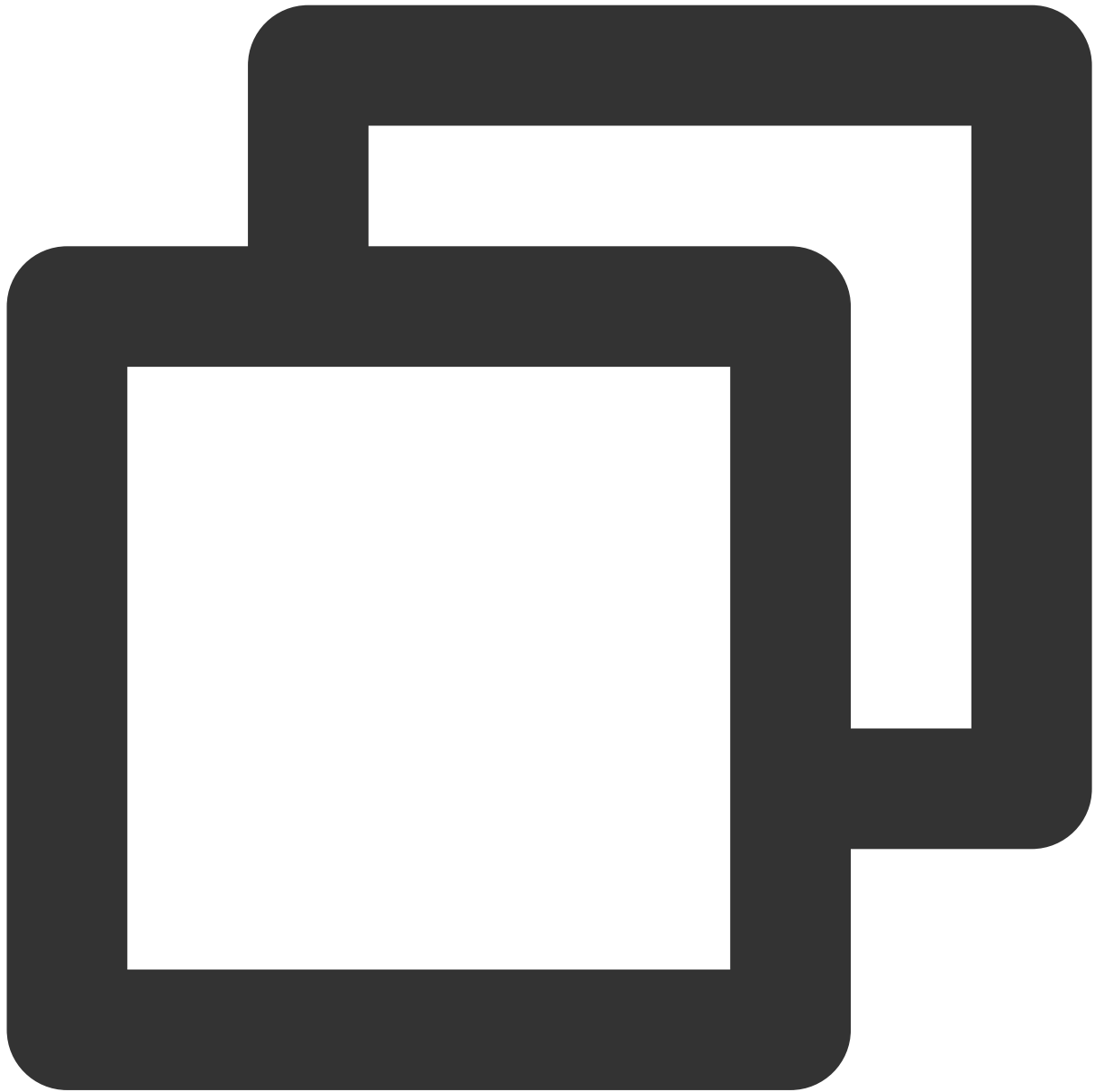
You have completed [Connecting Your Own Number](#). You have also finished the corresponding [IVR configuration](#).

Key Concepts

- SdkAppId:** The application ID users create on the [Cloud Contact Center console](#). You can create up to 20 Cloud Contact Center applications under one Tencent Cloud account, often starting with 140.
- UserID:** The account configured by the agent or administrator in the Cloud Contact Center, usually in the format of an email address. After the application is created for the first time, the main account can go to [Internal Message](#) (sub-account requires a subscription to Cloud Contact Center product messages) to view the contact center administrator account and password. Under one SDKAppID, multiple UserIDs can be configured. If the configuration limit is exceeded, more agents need to be purchased in [Agent Purchase](#).
- SecretId and SecretKey:** A certificate needed by developers to call cloud APIs, created on the [Tencent Cloud console](#).
- Token:** Login ticket, which is obtained by calling the Cloud API [CreateSDKLoginToken](#) to access. The correct approach is to place the token calculation code and encryption key on your business server, and then the app requests a token calculated in real time from your server when necessary.

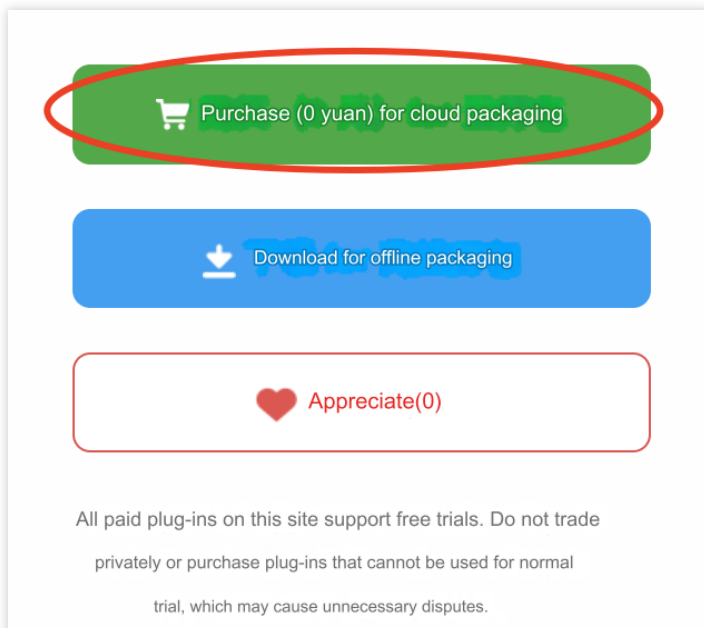
Integrating SDK

1. Integrate the TCCC SDK into your uni-app project using npm.



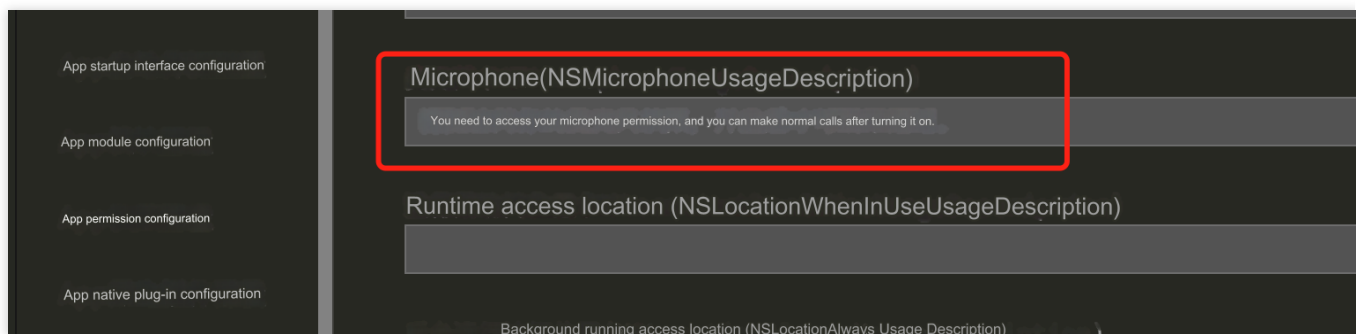
```
npm i tccc-sdk-uniapp
```

2. Purchase uni-app SDK plugin. Log in to [the uni-app native plugin marketplace](#) and make the purchase on the plugin details page (even free plugins can be purchased for 0 on the plugin marketplace). You can use the plugin in cloud packaging only after purchase. **When purchasing a plugin, select the right appid and bind the correct package name.**

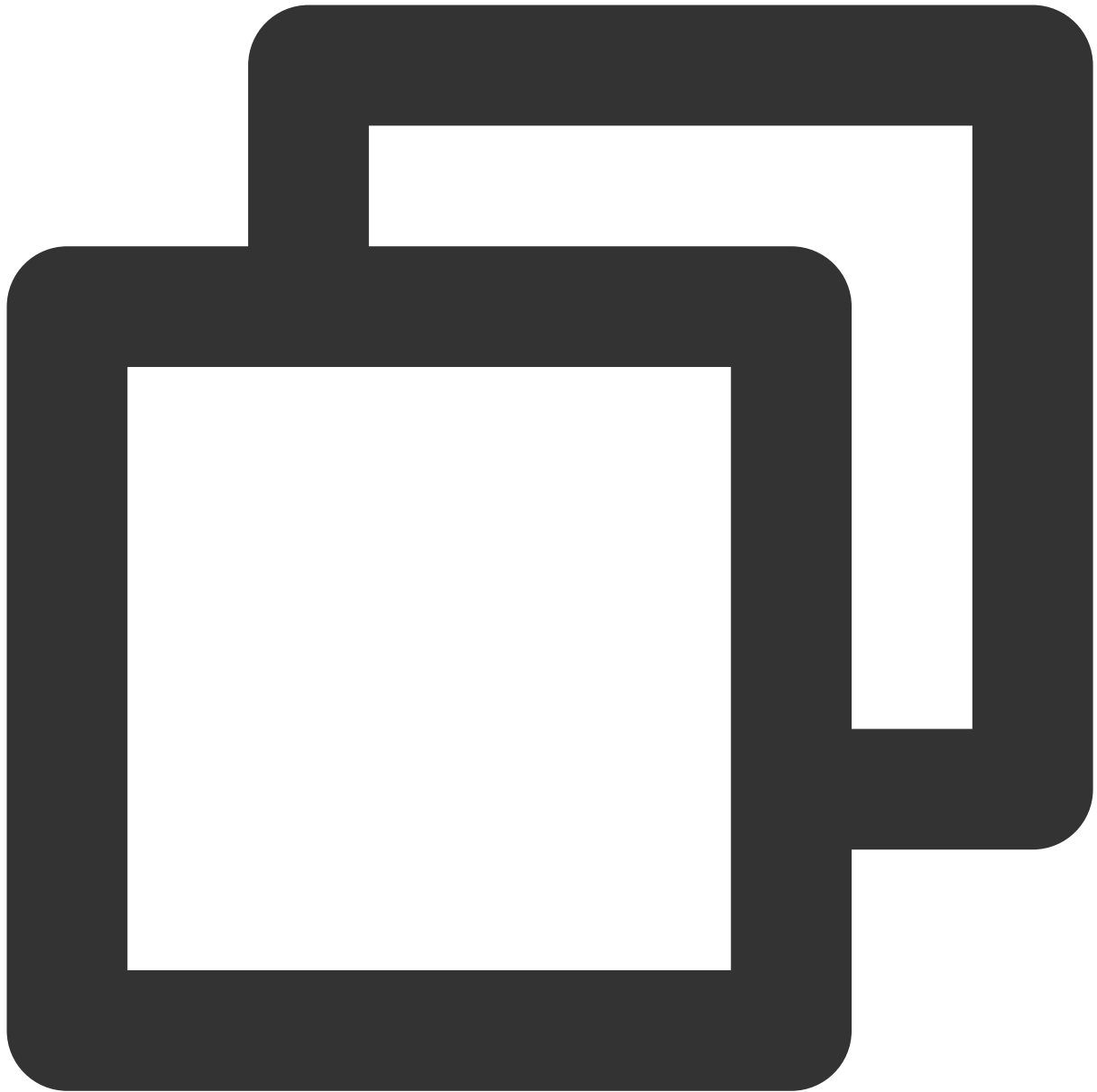


3. Configure permissions. Edit the **manifest.json** file to configure microphone permissions. The specifics are as follows:

The following permissions are needed on iOS: Privacy - Microphone Usage Description, and fill in the purpose of using the microphone.

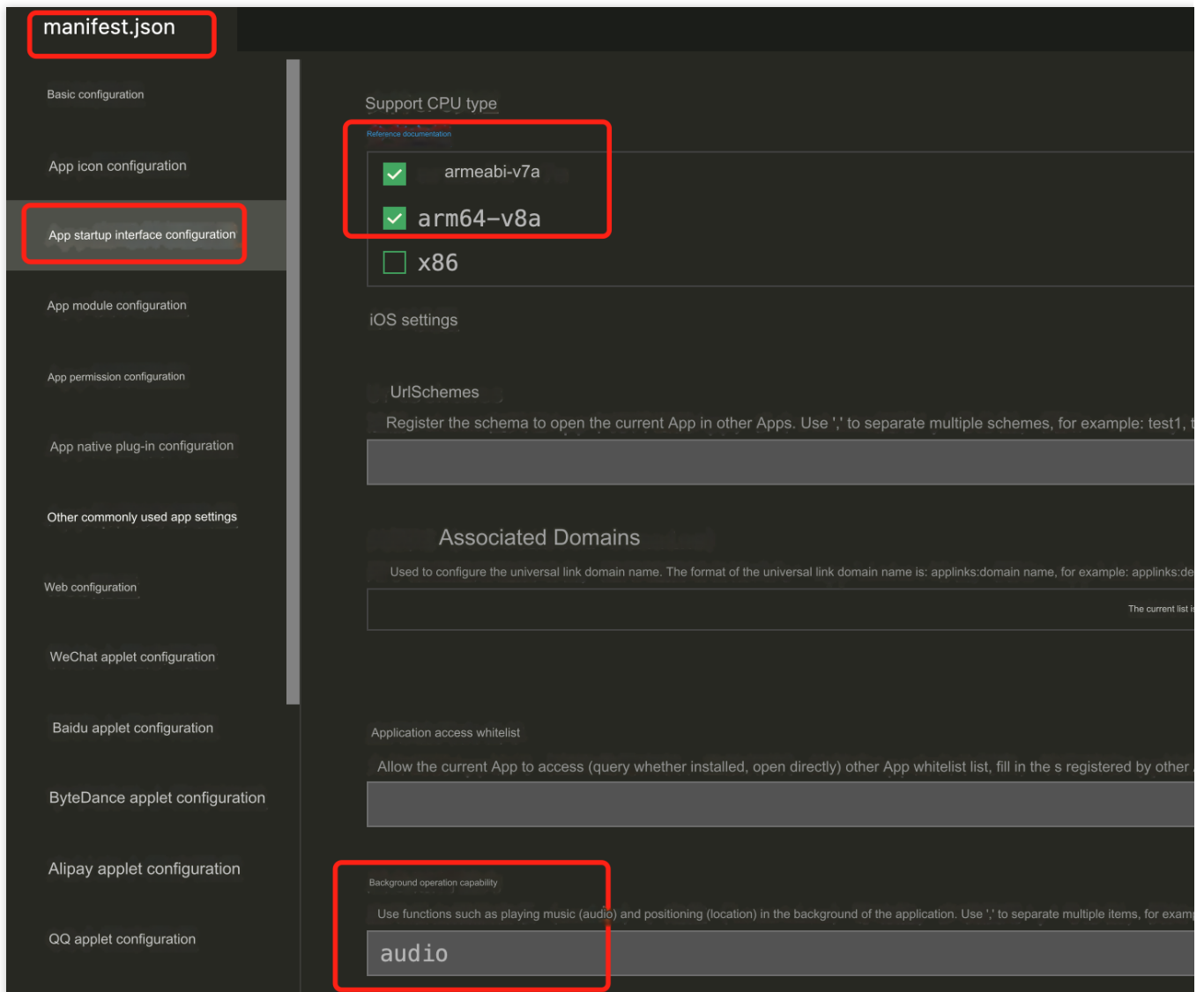


The following permissions are needed on Android:



```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

4. Configure audio to run in the background. When the mobile application is switched to the background, the operating system will pause the application's process to conserve resources. This means that all activities of the application will be stopped, including audio playback. On iOS, you need to configure **audio background mode** to ensure the application will not be terminated when the audio is being affected.



Note:

Without this permission, auto interrupts will occur when the call is switched to the background.

5. Use **Self-Defined Stand Packaging Run** (do not choose standard stand run) , and use **physical machine run** for the self-defined stand.

Run the project [tccc-workstation-uni-demo] to the Android device

refresh

✓ 7HX5T19925011835

☐ Runs with standard base

☒ Run using a custom base [What is a custom dock](#)

Package name: com.tencent.tccc.uniplugin.demo Modification time: 2023/4/11 16:56:05 uniRuntimeVersion: 3.7.3 Location

Troubleshooting Guide

run

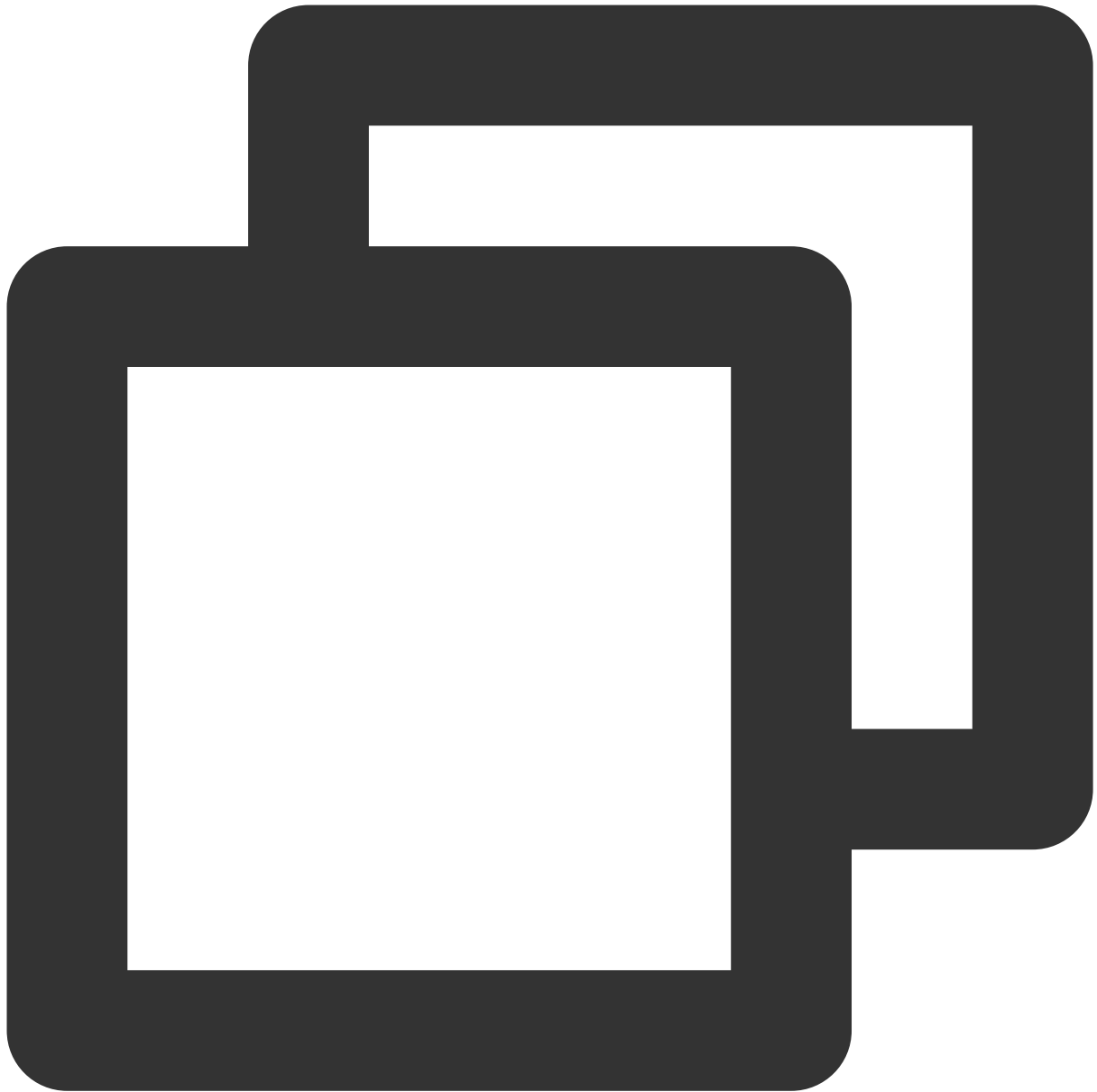
Note:

For details on a self-defined debugging stand and how to use it, please refer to [the official tutorial](#).

Code Implementation

For specific coding implementations, please refer to [API Overview and Examples](#).

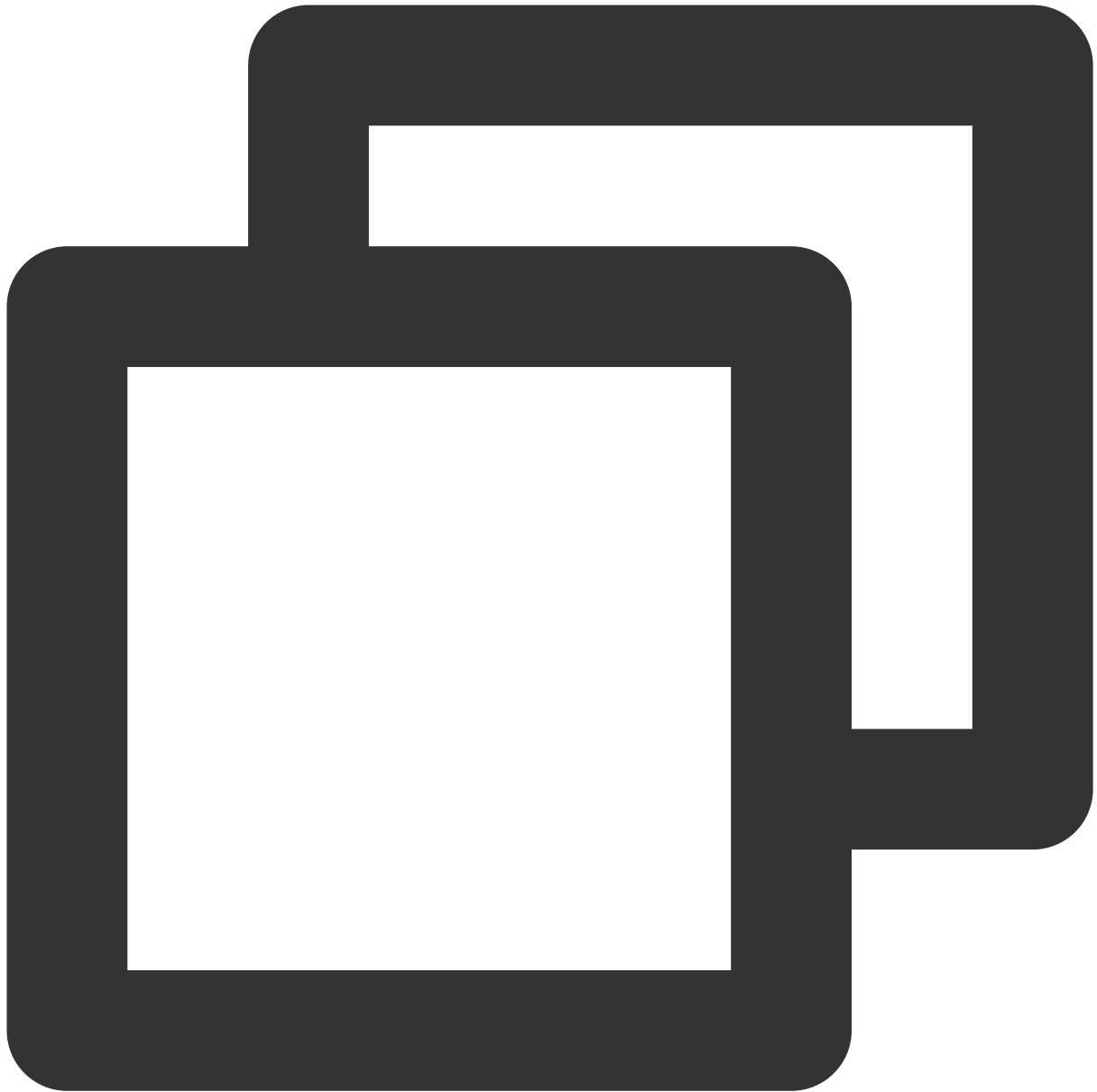
1. Create a TCCCWorkstation instance.



```
import {TcccWorkstation,TcccErrorCode} from "tccc-sdk-uniapp";
const tcccSDK = TcccWorkstation.sharedInstance();
// Listen to error events
tcccSDK.on("onError", (errCode,errMsg) => {

});
```

2. Log in.



```
const type = TCCCLoginType.Agent;
// For how to obtain sdkAppId, userId, and token, see the corresponding fields in K
// Agent login
tcccSDK.login({
  sdkAppID: 1400000000,    // Replace it with your own SdkAppId
  userId: "xxx@qq.com", // Replace it with the agent account
  token: "xxxx", // Replace it with the token obtained through cloud API CreateSD
  type: type,
}, (code,message) => {
  if (code == TcccErrorCode.ERR_NONE) {
    // Login succeeded
  }
})
```

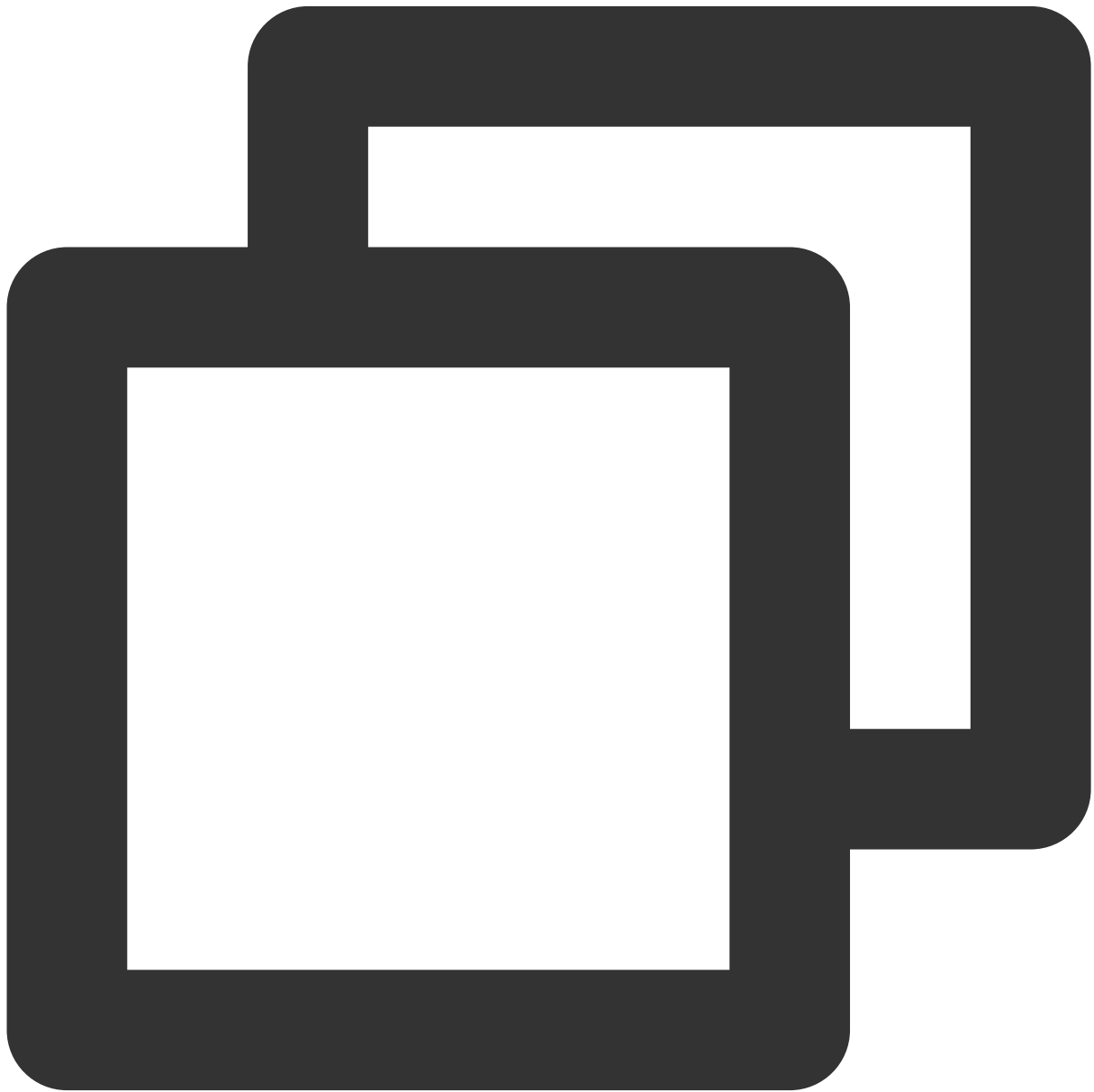


```
    } else {  
        // Login failed  
    }  
    });
```

Note:

To obtain the token, backend development is required, and you need to call the Cloud API [CreateSDKLoginToken](#) to access.

3. Initiate a call.



```
// Initiate a call
```

```
tcccSDK.call({
  to: '134xxxx',          // Contact number (required)
  remark: "xxx",          // Number remarks, which will replace the number displayed
  uui: "xxxx",            // User-defined data (optional)
}, (code,message) => {
  if (code == TcccErrorCode.ERR_NONE) {
    // Initiation succeeded
  } else {
    // Initiation failed
  }
});
```

4. Handle the callback of the correspondent's answer.



```
tcccSDK.on('onAccepted', (sessionId) => {  
    // The correspondent has answered  
});
```

5. End the call.



```
// End the call  
tcccSDK.terminate();
```

Android

Last updated : 2024-04-01 17:43:49

Quickly Integrating Cloud Contact Center Android SDK

This topic mainly introduces how to quickly integrate Cloud Contact Center Android SDK into your project. Just follow the steps for configuration, and you can complete the integration of the SDK.

Environment Requirements

Android Studio 3.5+.

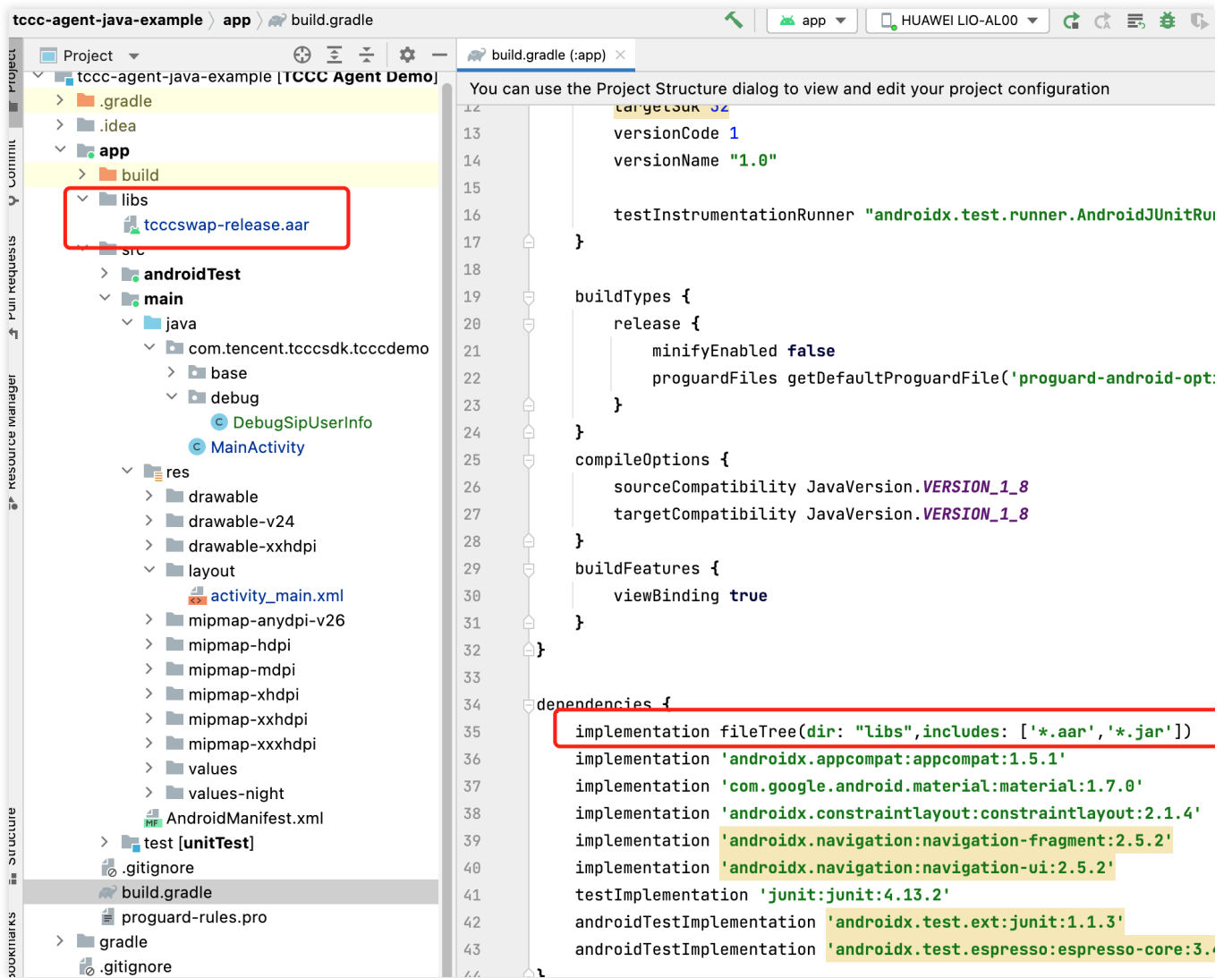
Android 4.1 (SDK API 16) or later.

Integrating SDK (aar, jar)

Manual Download (aar, jar)

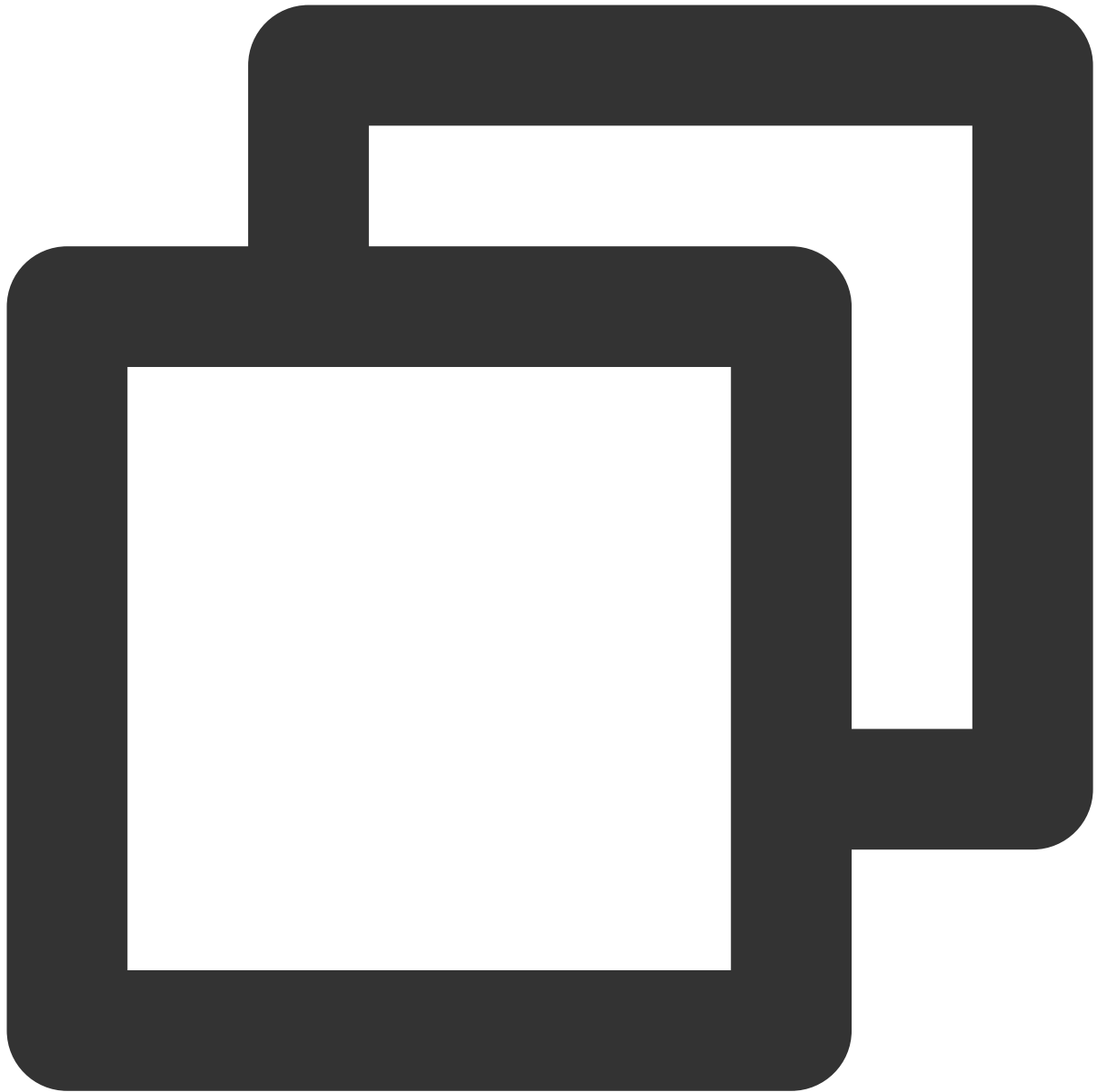
Currently we have not released to mavenCentral, so you can only manually download SDK into your project:

1. Download the latest version of [TCCC Agent SDK](#).
2. Copy the downloaded aar file into the **app/libs** directory of your project.
3. Specify the local repository path in build.gradle in the root directory of your project.



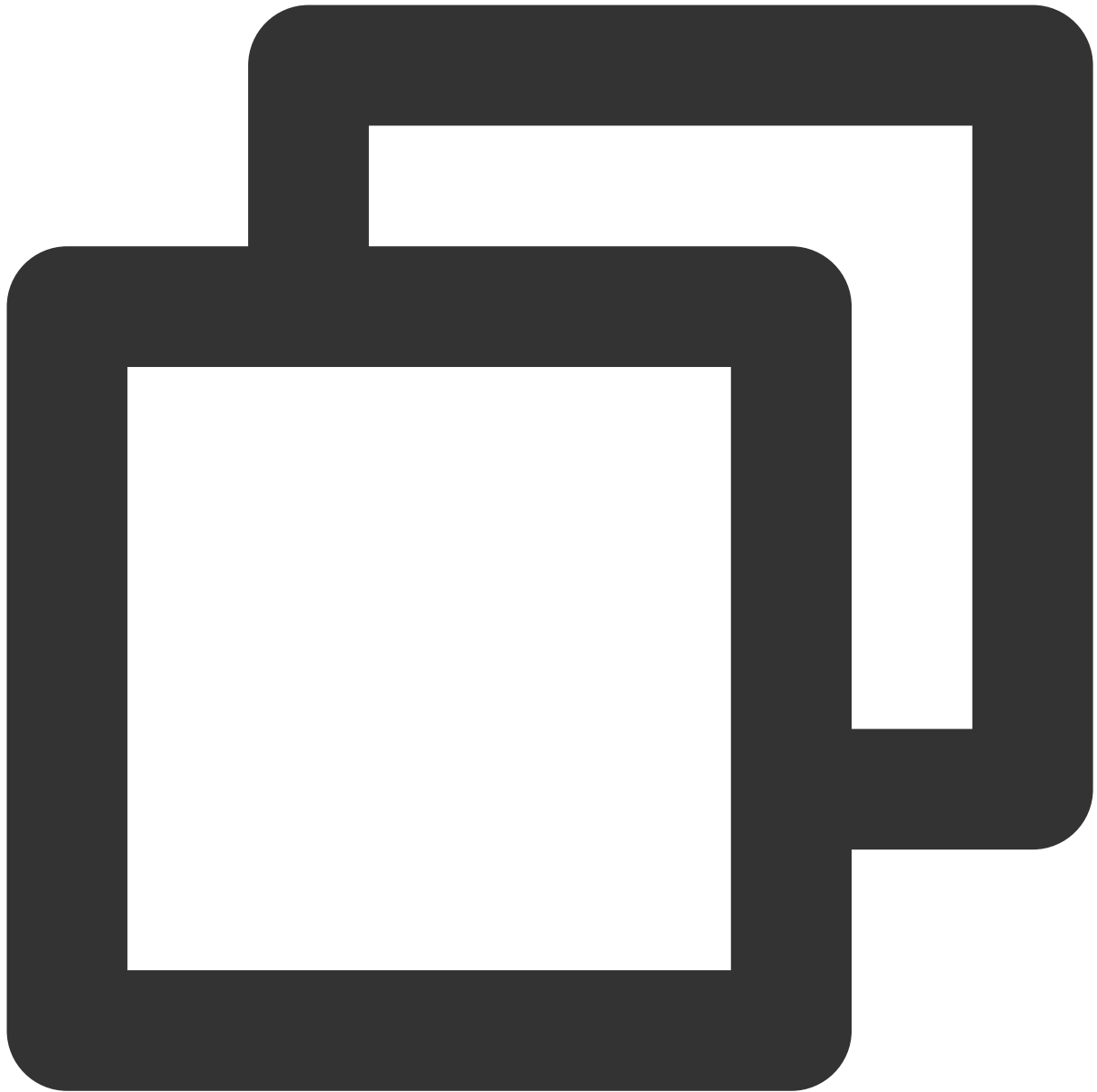
The screenshot displays an IDE interface for a project named "tccc-agent-java-example". The left sidebar shows the project structure, with the "libs" folder and the "tccswap-release.aar" file highlighted. The main editor shows the "build.gradle" file for the "app" module. The file contains configuration for the Android build system, including version codes, names, instrumentation runners, build types, compile options, build features, and dependencies. The "dependencies" section is highlighted, showing the inclusion of the "tccswap-release.aar" file and various Android dependencies.

```
12 targetSdkVersion 32
13 versionCode 1
14 versionName "1.0"
15
16 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
17
18
19 buildTypes {
20     release {
21         minifyEnabled false
22         proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
23     }
24 }
25 compileOptions {
26     sourceCompatibility JavaVersion.VERSION_1_8
27     targetCompatibility JavaVersion.VERSION_1_8
28 }
29 buildFeatures {
30     viewBinding true
31 }
32
33
34 dependencies {
35     implementation fileTree(dir: "libs", includes: ['*.aar', '*.jar'])
36     implementation 'androidx.appcompat:appcompat:1.5.1'
37     implementation 'com.google.android.material:material:1.7.0'
38     implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
39     implementation 'androidx.navigation:navigation-fragment:2.5.2'
40     implementation 'androidx.navigation:navigation-ui:2.5.2'
41     testImplementation 'junit:junit:4.13.2'
42     androidTestImplementation 'androidx.test.ext:junit:1.1.3'
43     androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.0'
```



```
implementation fileTree(dir: "libs",includes: ['*.aar','*.jar'])
```

4. Specify the CPU architecture used by the app in defaultConfig in app/build.gradle.

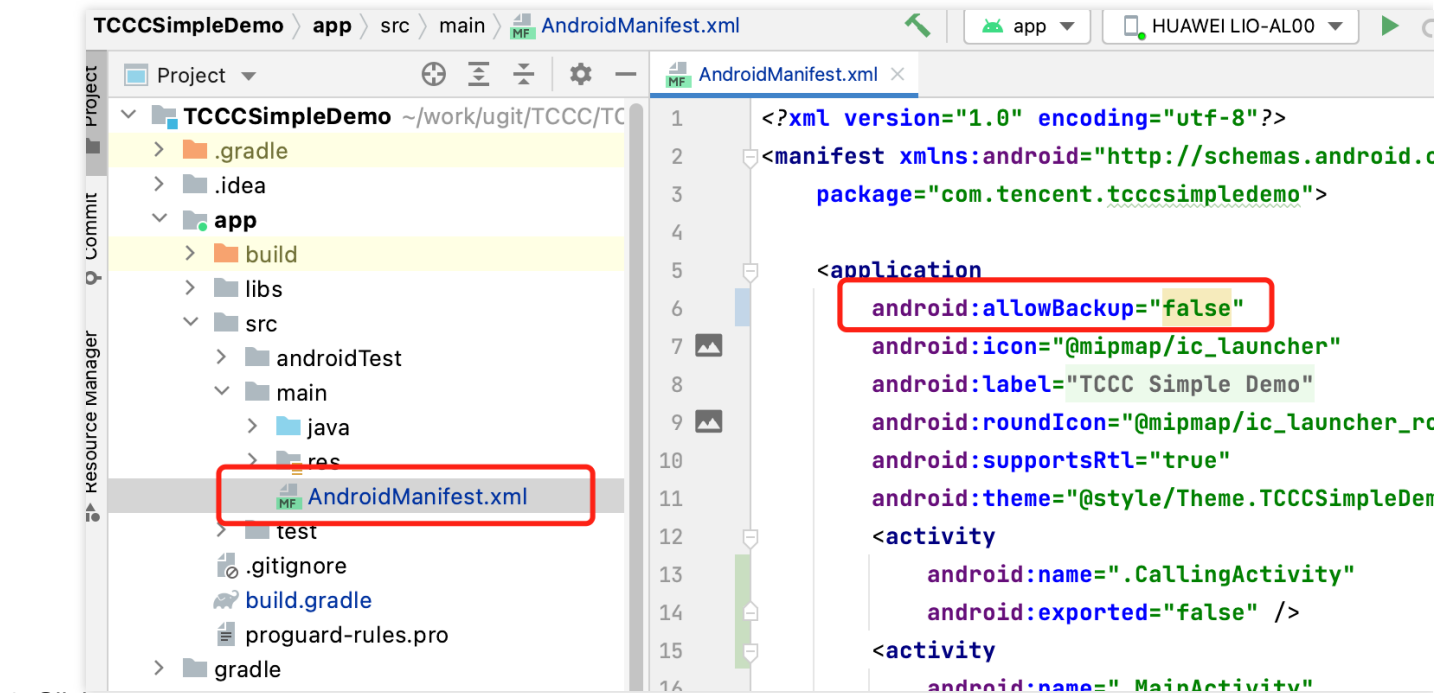


```
defaultConfig {  
    ndk {  
        abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"  
    }  
}
```

Note:

Currently, the TCCC Agent SDK supports armeabi, armeabi-v7a, and arm64-v8a.

5. In `app/src/AndroidManifest.xml`, specify that the app does not permit the application to be involved in the backup and restore infrastructure.



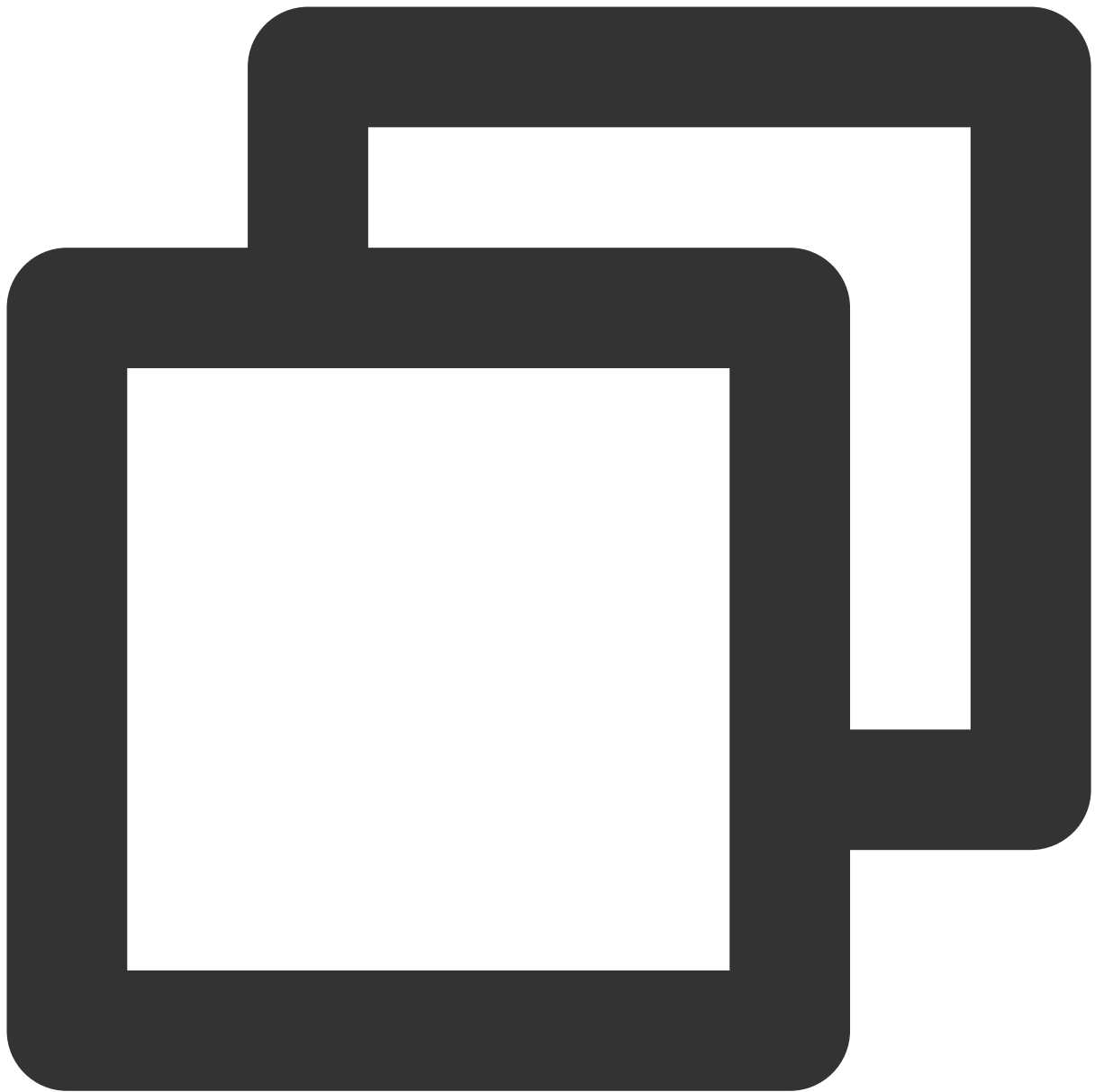
6. Click



Sync Now to complete the integration of the TCCC Agent SDK.

Configuring Permissions

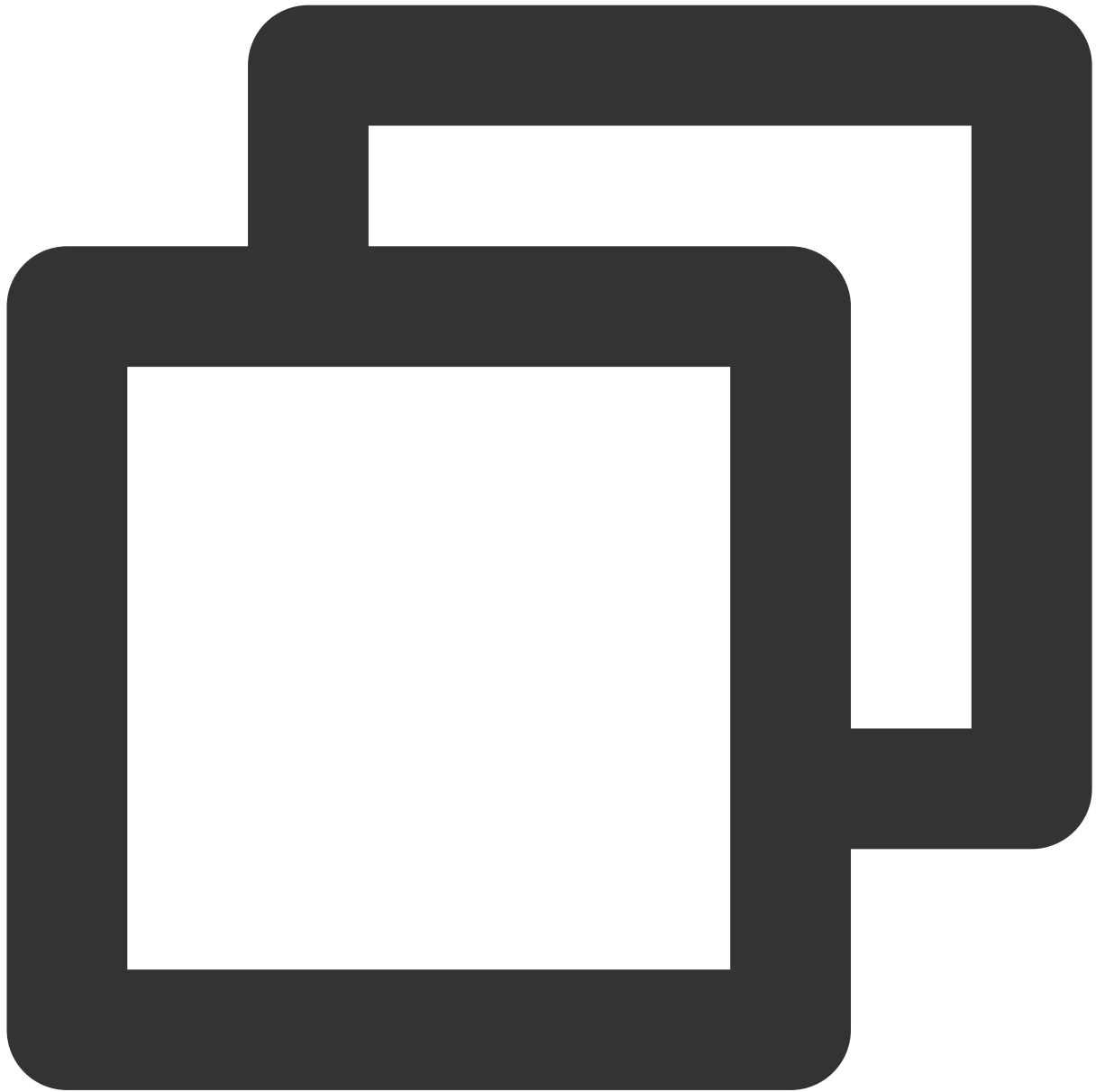
In AndroidManifest.xml, set the permissions of the app. The TCCC Agent SDK needs the following permissions:



```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

Setting Obfuscation Rules

In the `proguard-rules.pro` file, add the related classes of the TCCC SDK to the non-obfuscated list:



```
-keep class com.tencent.** { *; }
```

Code Implementation

For specific coding implementation, please refer to [Android SDK API](#).

iOS

Last updated : 2024-04-02 10:52:32

Integrating Cloud Contact Center iOS Agent SDK

This topic mainly introduces how to quickly integrate Cloud Contact Center iOS Agent SDK into your project. Just follow the steps for configuration, and you can complete the integration of the SDK.

Environment Requirements

Xcode 9.0+.

A real iPhone or iPad running iOS 9.0 or later.

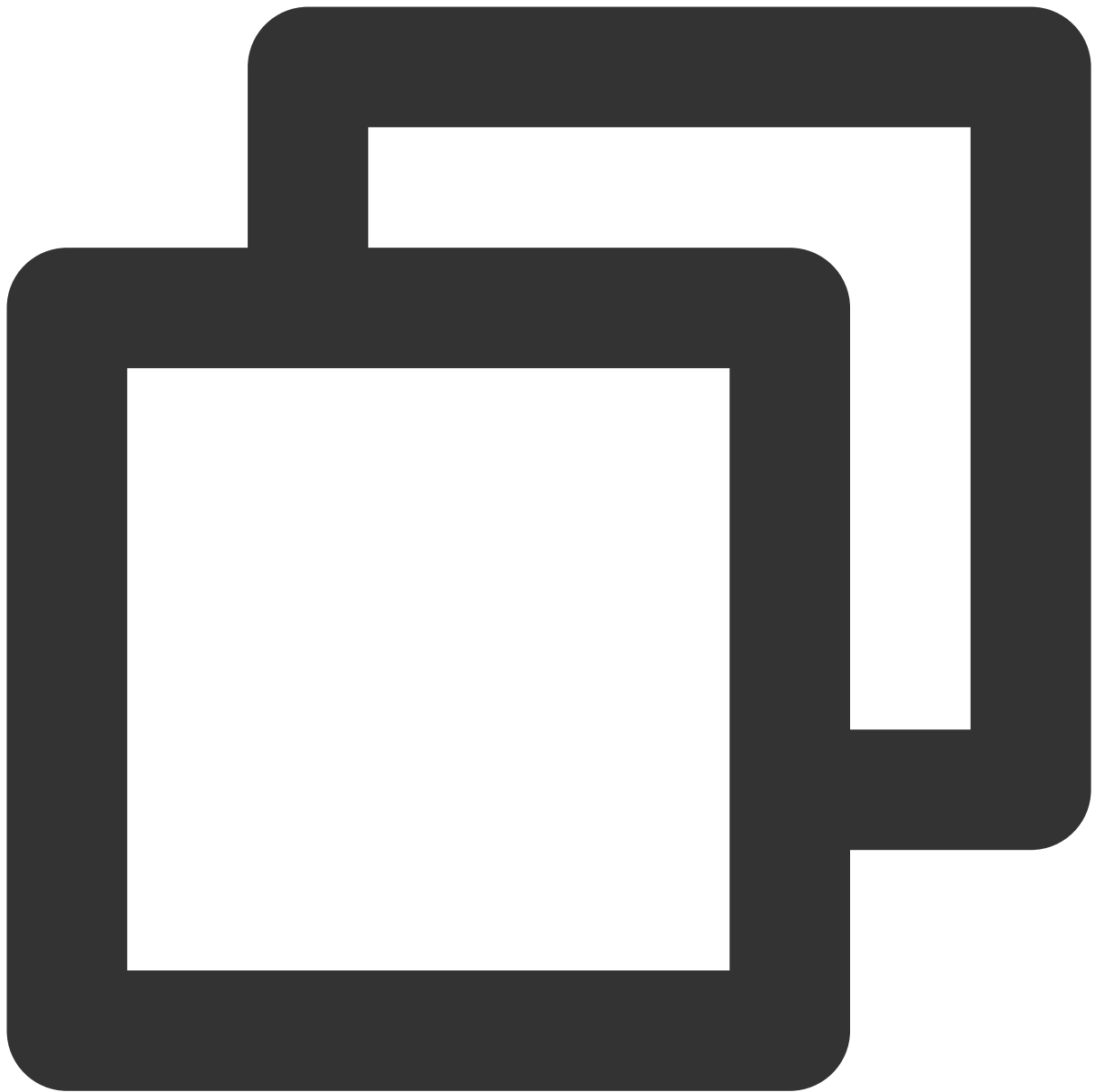
The project has been configured with a valid developer signature.

Integrating SDK

Solution 1: Using CocoaPods

1. Install CocoaPods.

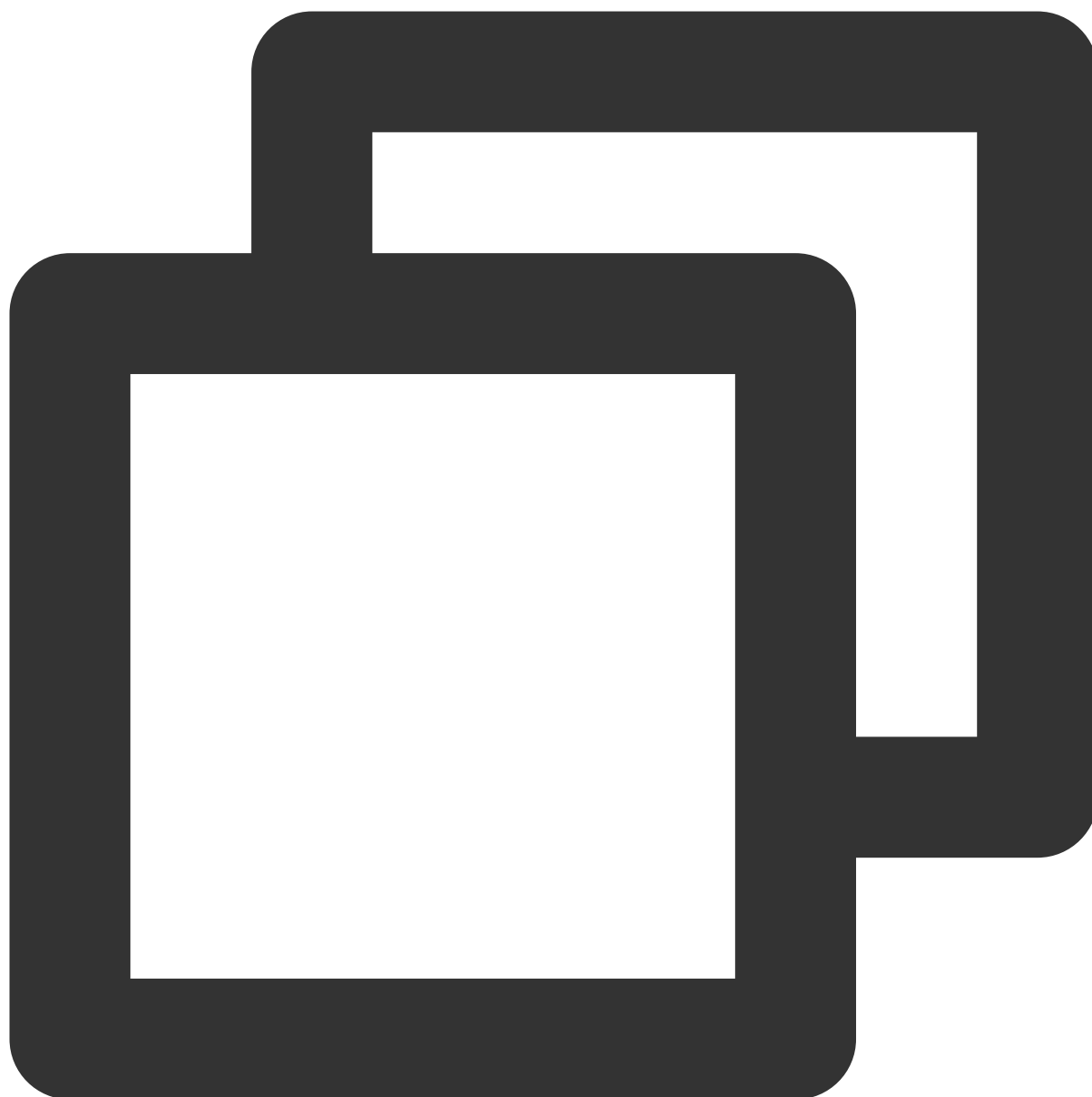
Enter the following command in a terminal window (you need to install Ruby on your Mac first):



```
sudo gem install cocoapods
```

2. Create a Podfile.

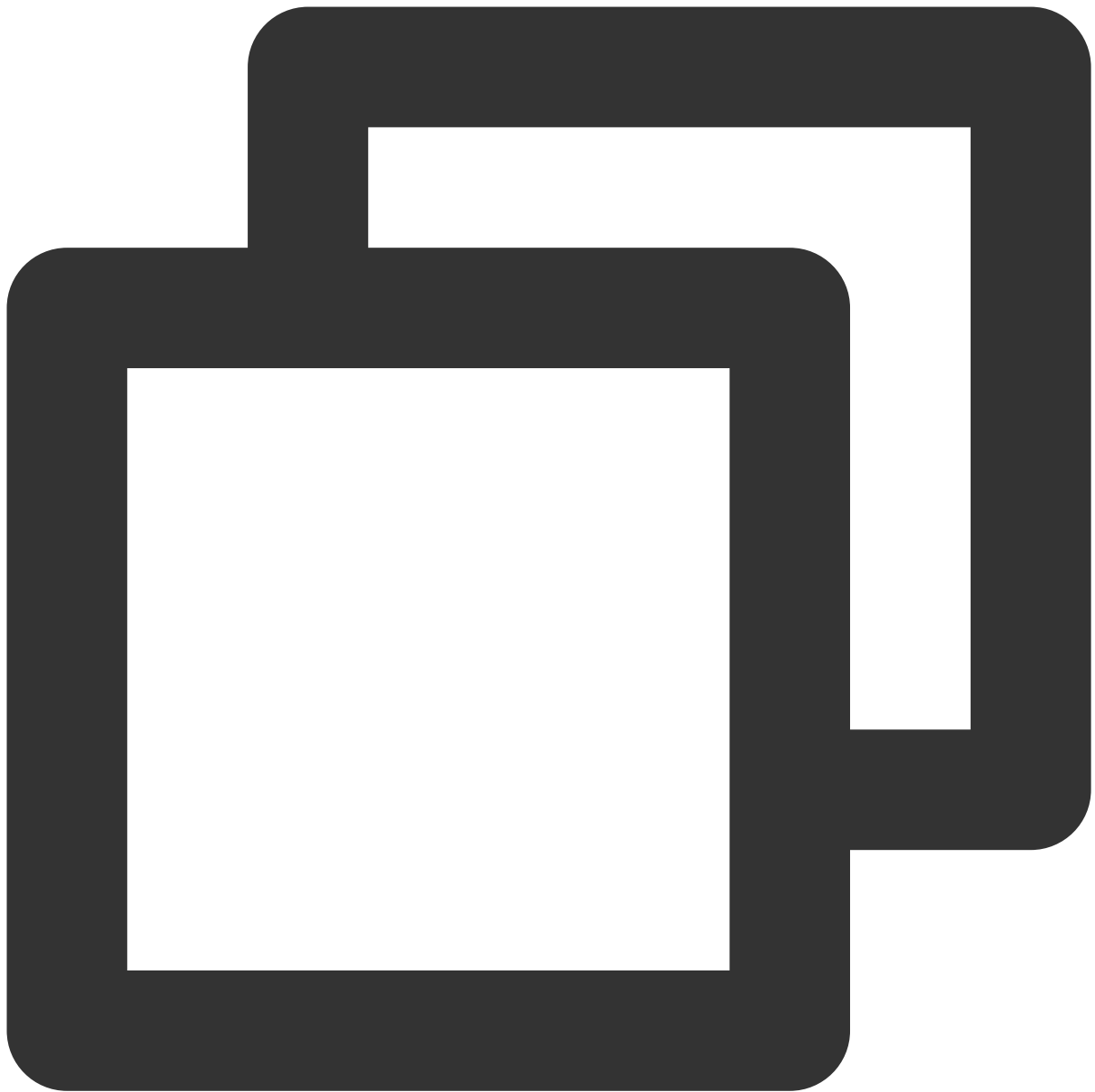
Go to the directory of your project and enter the following command to create a Podfile in the directory.



```
pod init
```

3. Edit the Podfile

Edit the Podfile according to your project needs:

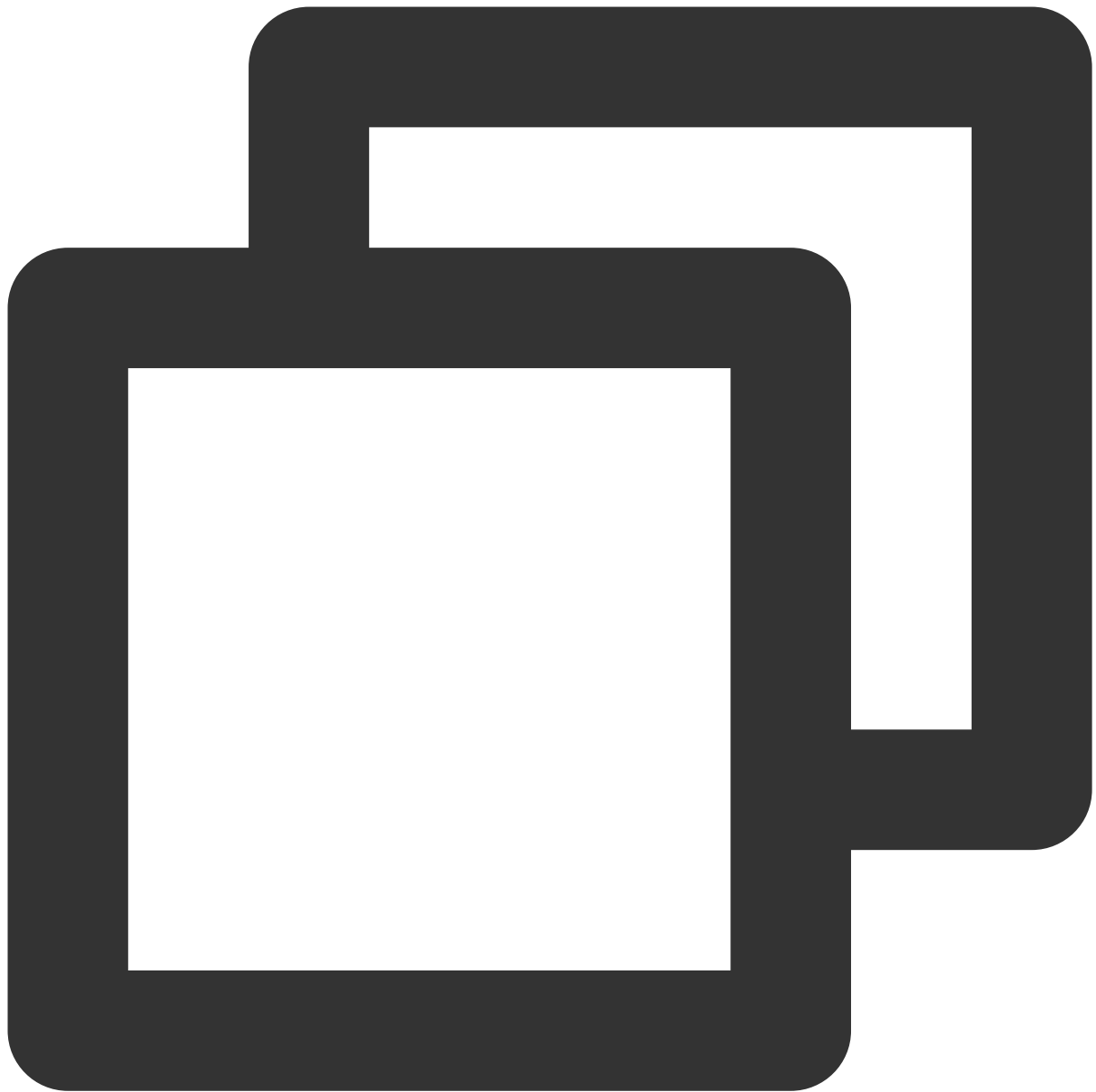


```
platform :ios, '11.0'

target 'App' do
  pod 'TCCCSDK_Ios', :podspec => 'https://tccc.qcloud.com/assets/doc/Agent/CppSDK'
end
```

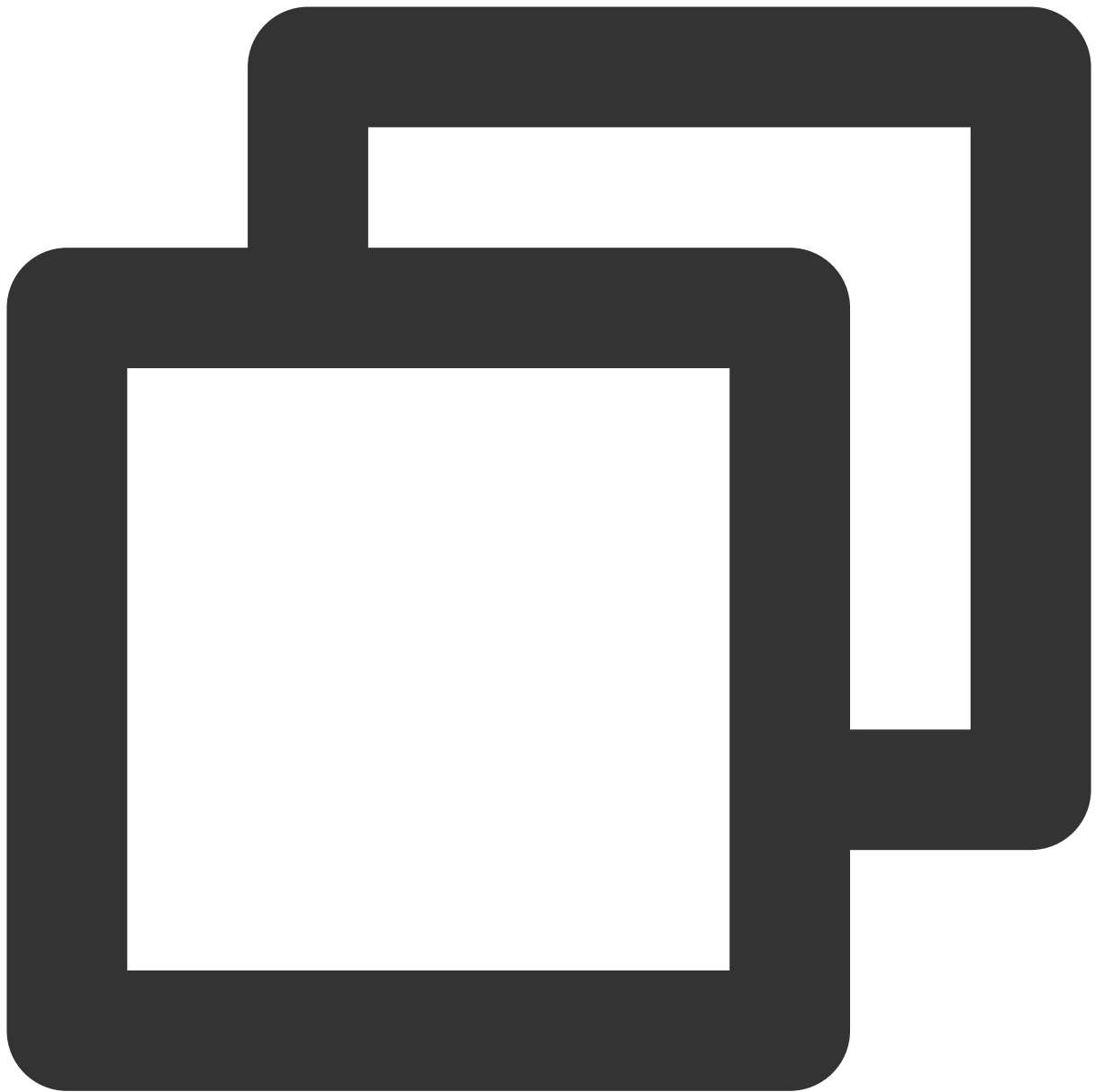
4. Update the local repository and install the SDK

Enter the following command in the Terminal window to update the local library file and install the SDK:



```
pod install
```

Alternatively, run the following command to update the local repository:

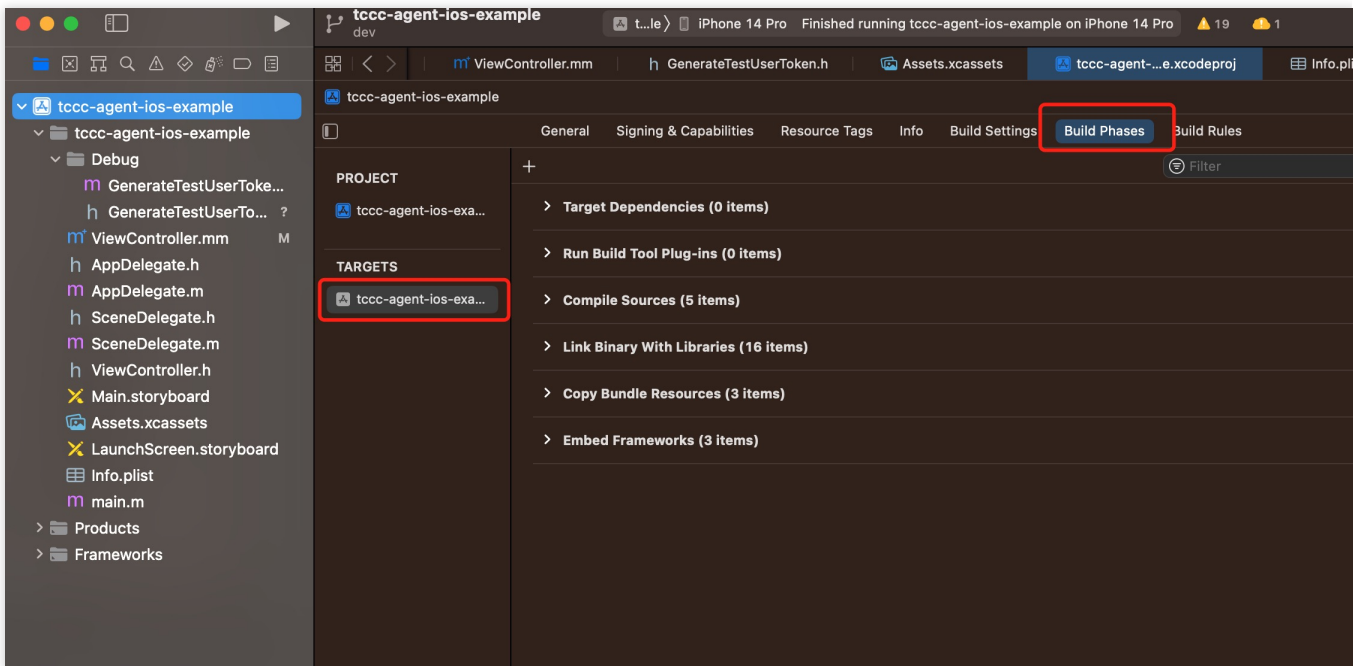


```
pod update
```

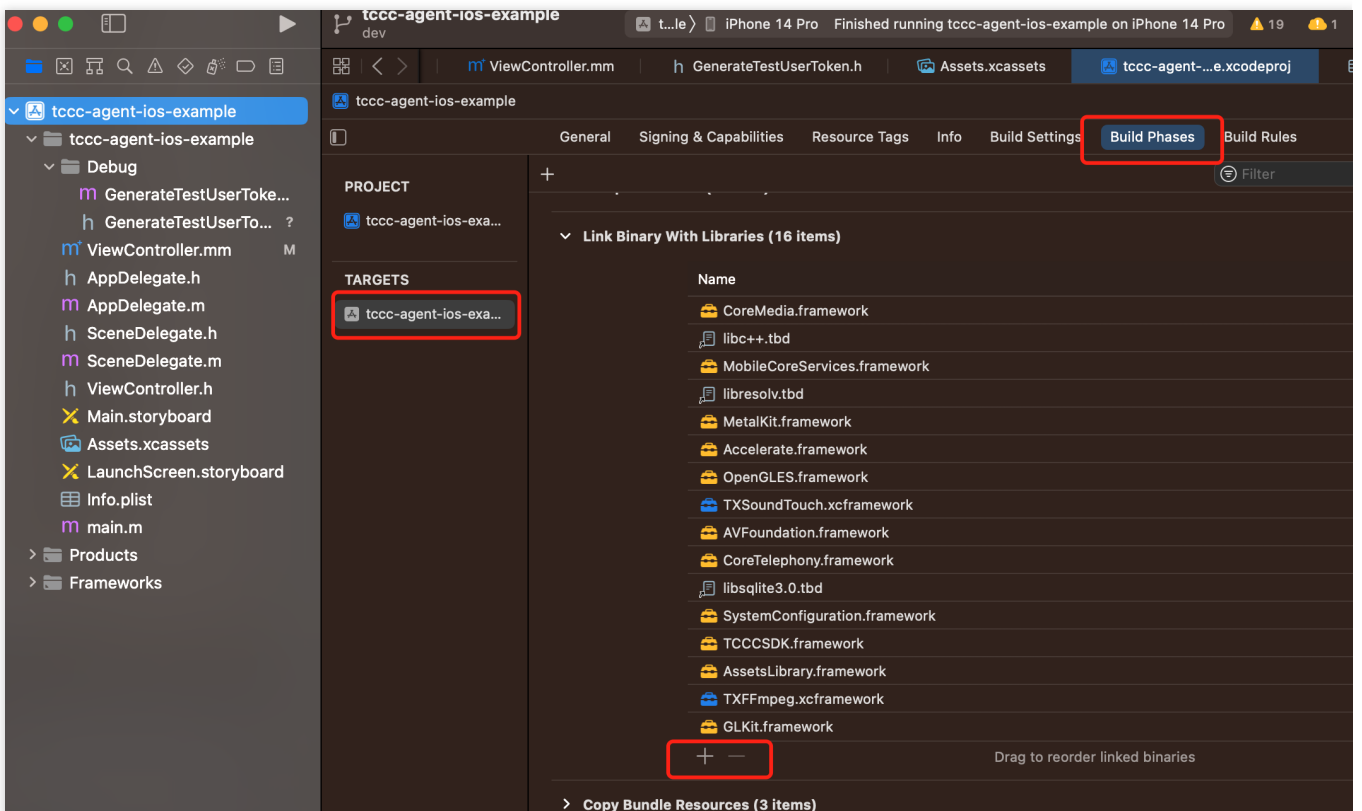
An XCWORKSPACE project file integrated with the SDK will be generated. Double-click to open it.

Solution 2: Manual Download

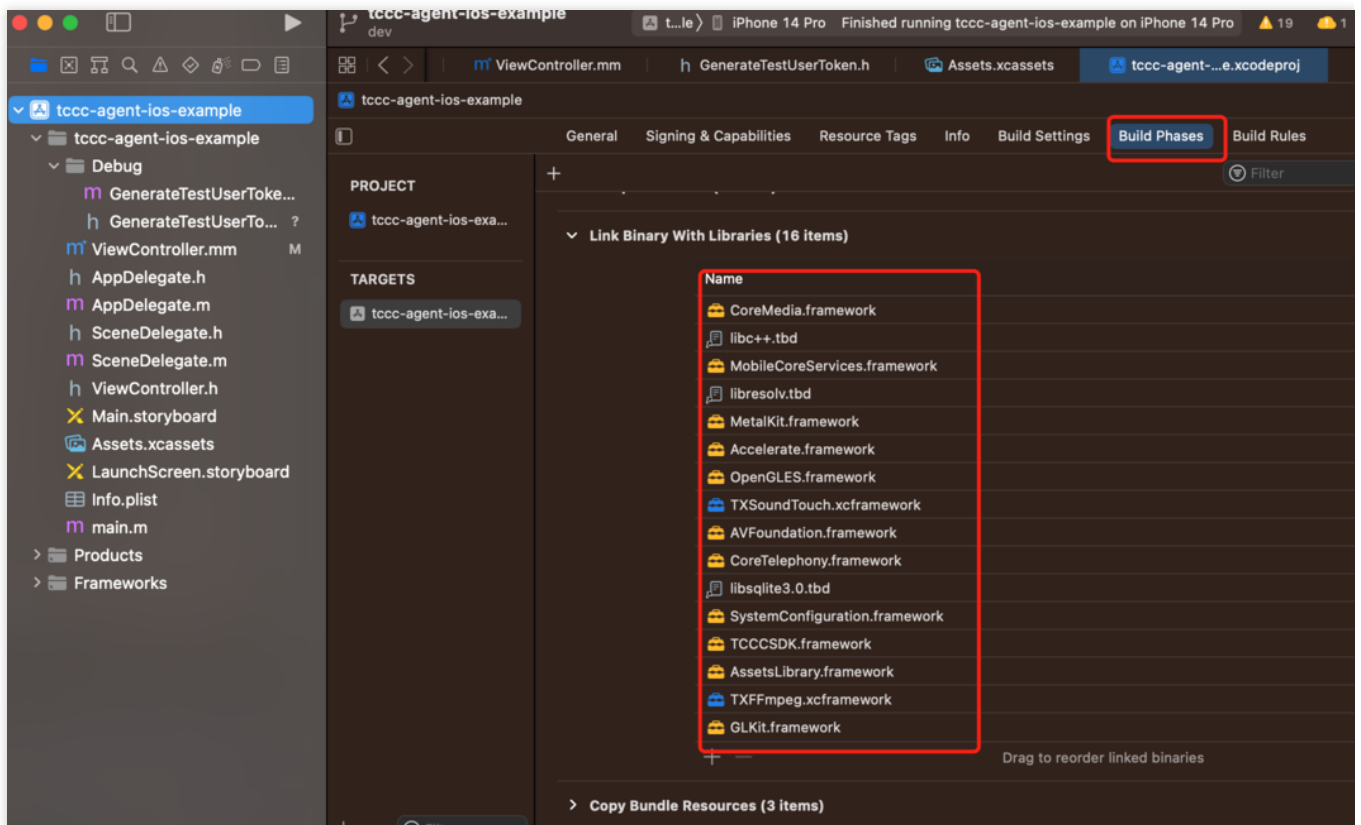
1. Download the latest version of [TCCC Agent SDK](#).
2. Open your Xcode project, select the target to run, and click **Build Phases**.



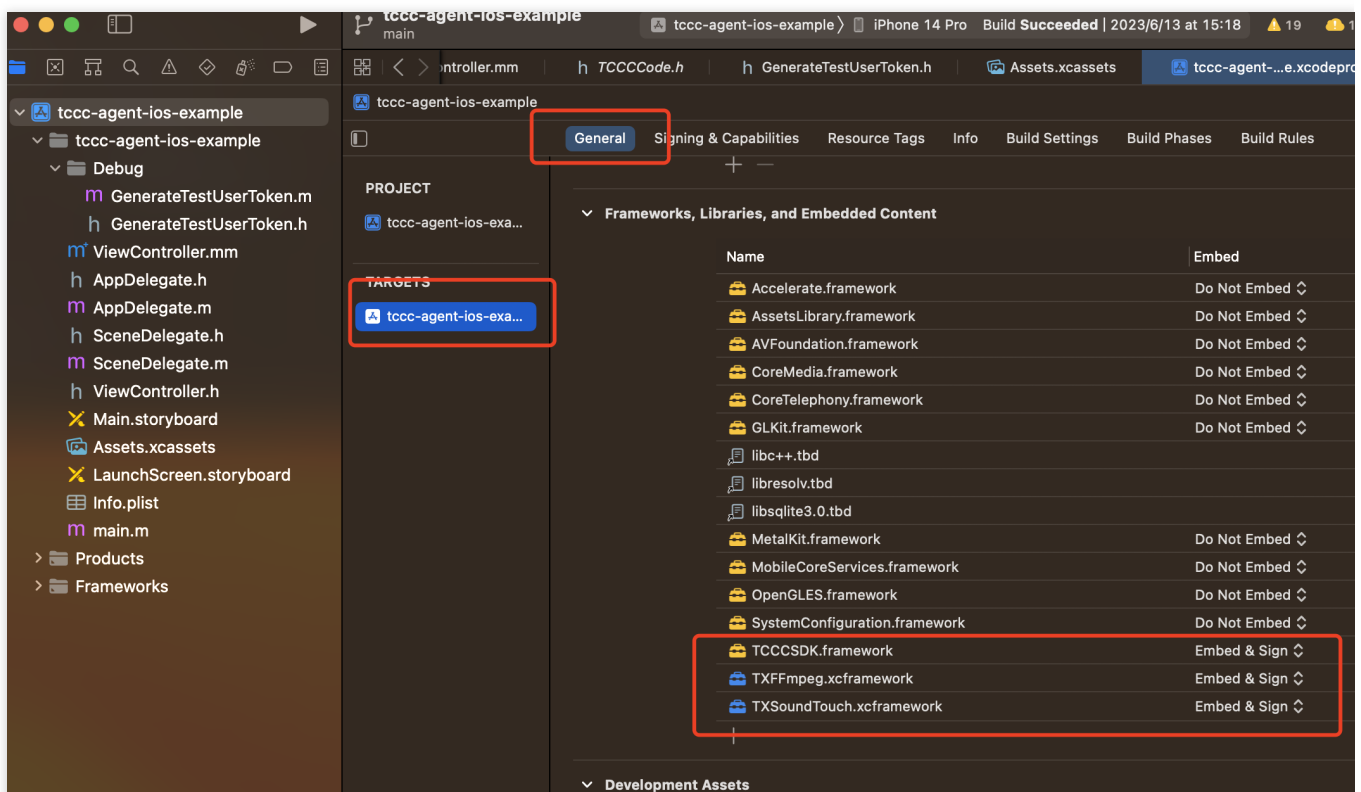
3. Click **Link Binary with Libraries** to expand, and click the "+" icon below to add dependency libraries.



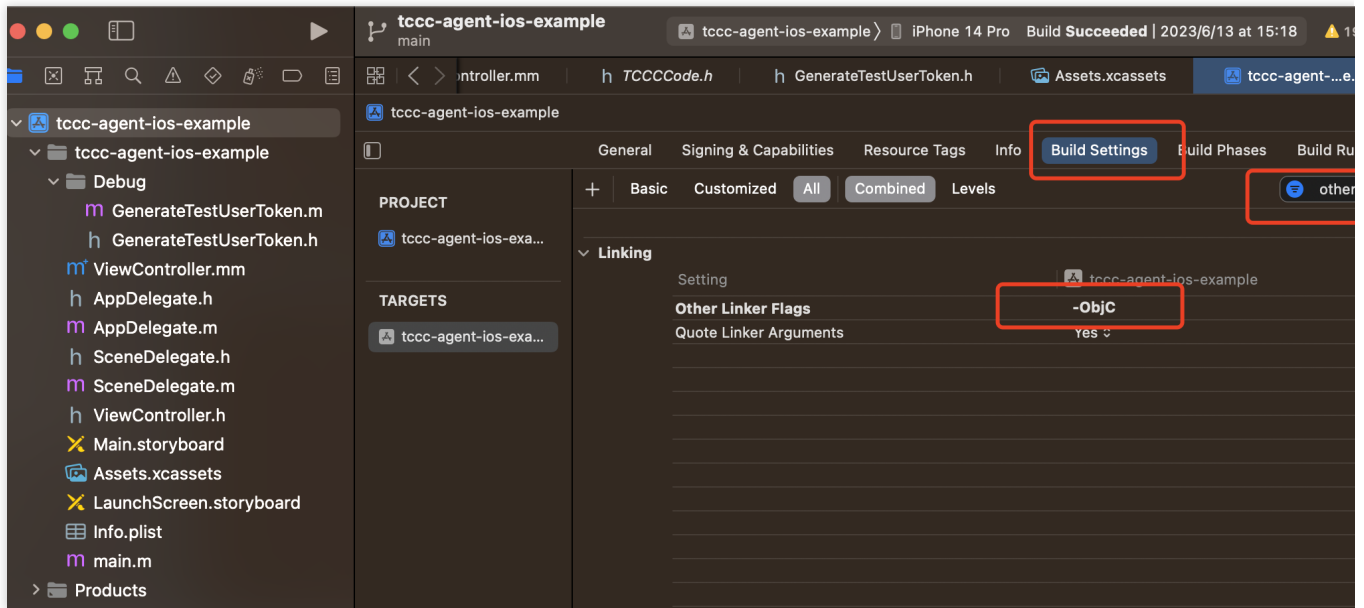
4. Add the downloaded **TCCSDK.framework**, **TXFFmpeg.xcframework**, and **TXSoundTouch.xcframework**, and the required dependency libraries **GLKit.framework**, **AssetsLibrary.framework**, **SystemConfiguration.framework**, **libsqlite3.0.tbd**, **CoreTelephony.framework**, **AVFoundation.framework**, **OpenGL.framework**, **Accelerate.framework**, **MetalKit.framework**, **libresolv.tbd**, **MobileCoreServices.framework**, **libc++.tbd**, and **CoreMedia.framework**.



5. Click **General**, select **Frameworks, Libraries, and Embedded Content**. Check whether the dynamic libraries **TCCSDK.framework**, **TXFFmpeg.xcframework**, and **TXSoundTouch.xcframework** have been added, and whether **Embed & Sign** is correctly selected. If not, click the "+" icon below to add them in order.

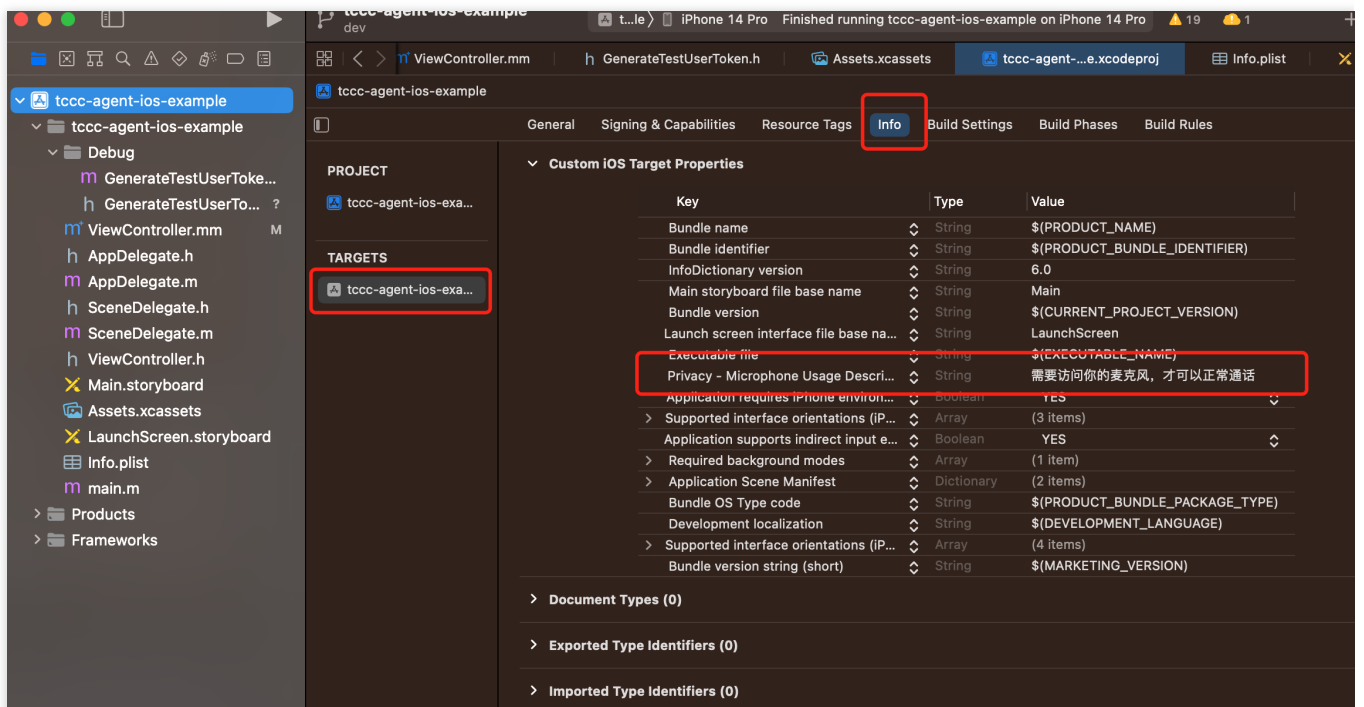


6. Add **-ObjC** configuration in **Other Linker Flags** of the project target Build Settings.

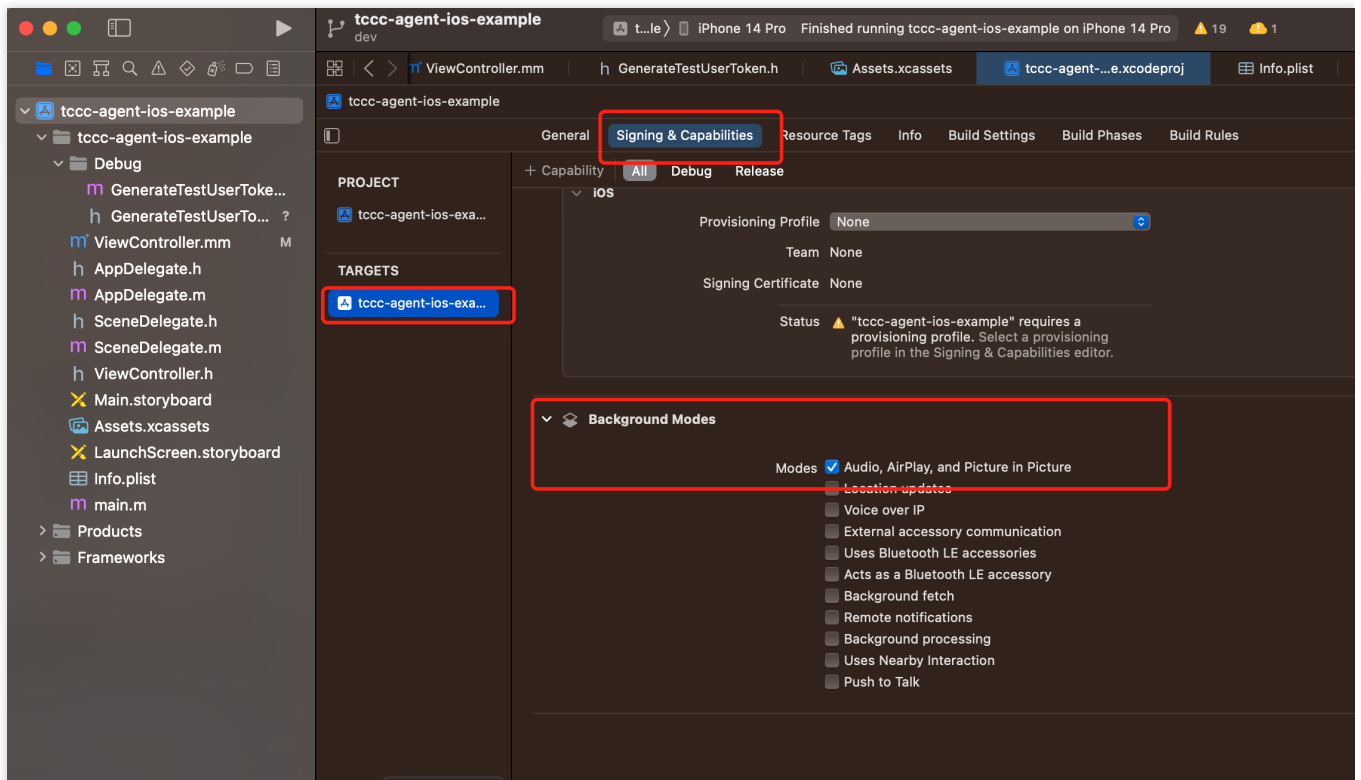


Configuring Permissions

1. If you need to use the audio and video features provided by the SDK, you need to authorize the use of the microphone for the app. Add the corresponding microphone prompt information when the system pops up the authorization dialog box in the Info.plist of the app.



2. If you need the app to continue running related features in the background, you can select the current project in Xcode, set Background Modes in Capabilities to ON, and select Audio, AirPlay and Picture in Picture, as shown below:



Code Implementation

We currently provide Swift, OC, and C++ interfaces for developers to choose from. You can use the following code to import the header file:

Swift

Objective-C

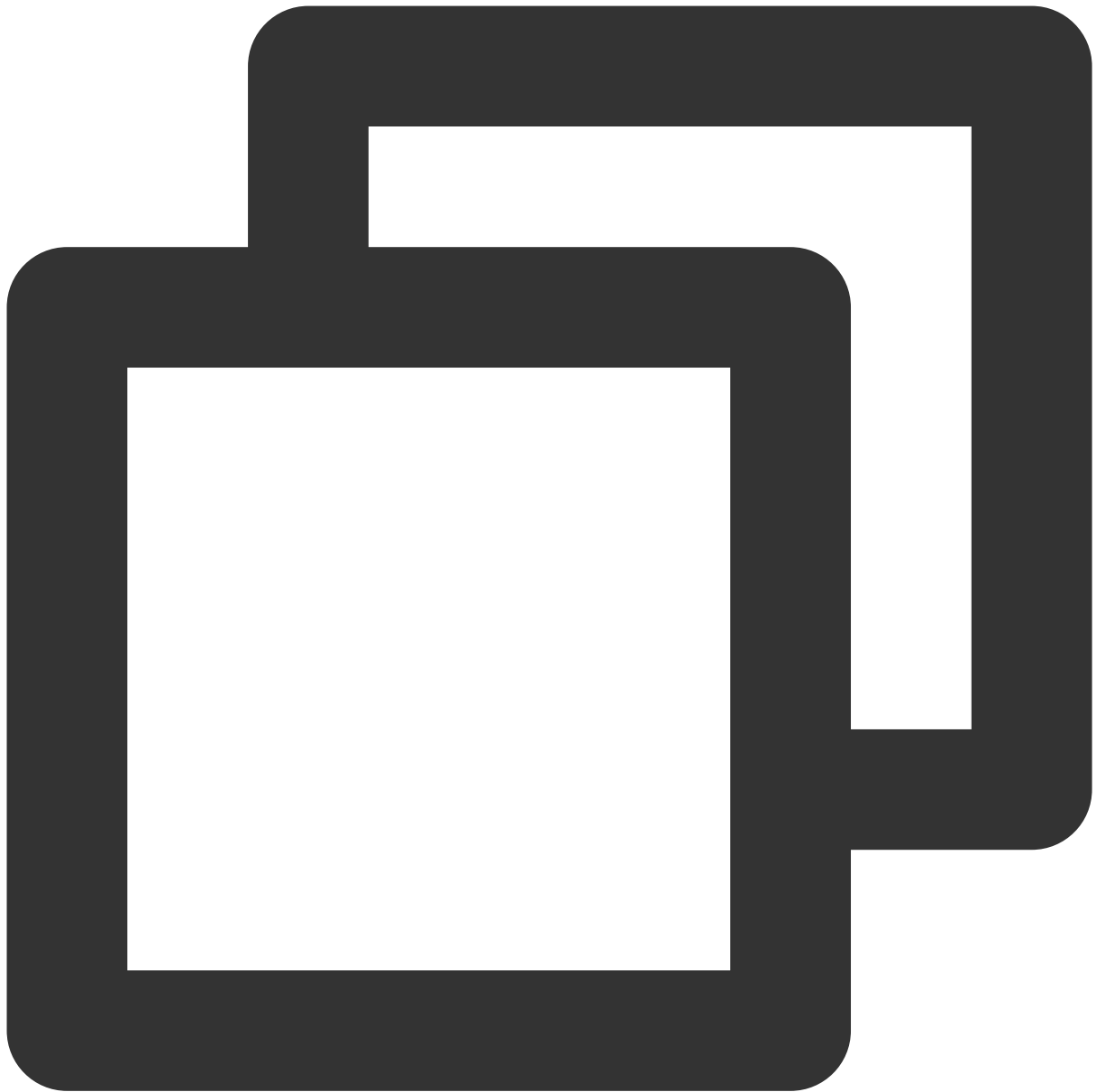
C++



```
import TCCCSDK
// Obtain the tcccSDK singleton
let tcccSDK: TCCCWorkstation = {
    return TCCCWorkstation.sharedInstance()
}()
// Obtain SDK version number
let version = TCCCWorkstation.getSDKVersion()
```



```
// Import the OC header file
#import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"
// Obtain the tcccSDK singleton
- (TCCCWorkstation*)tcccSDK {
    if (!_tcccSDK) {
        _tcccSDK = [TCCCWorkstation sharedInstance];
    }
    return _tcccSDK;
}
// Obtain SDK version number
NSString* version = [TCCCWorkstation getSDKVersion];
```

```
// Import the C++ header file
#include "TCCSDK/tccc/include/ITCCCWorkstation.h"
// Use the tccc namespace
using namespace tccc;
// Obtain the tcccSDK singleton
ITCCCWorkstation* tcccSDK = getTCCCShareInstance();
// Obtain SDK version number
const char * version = tcccSDK->getSDKVersion();
```

For specific coding implementations, please refer to [API Overview and Examples](#).

FAQs

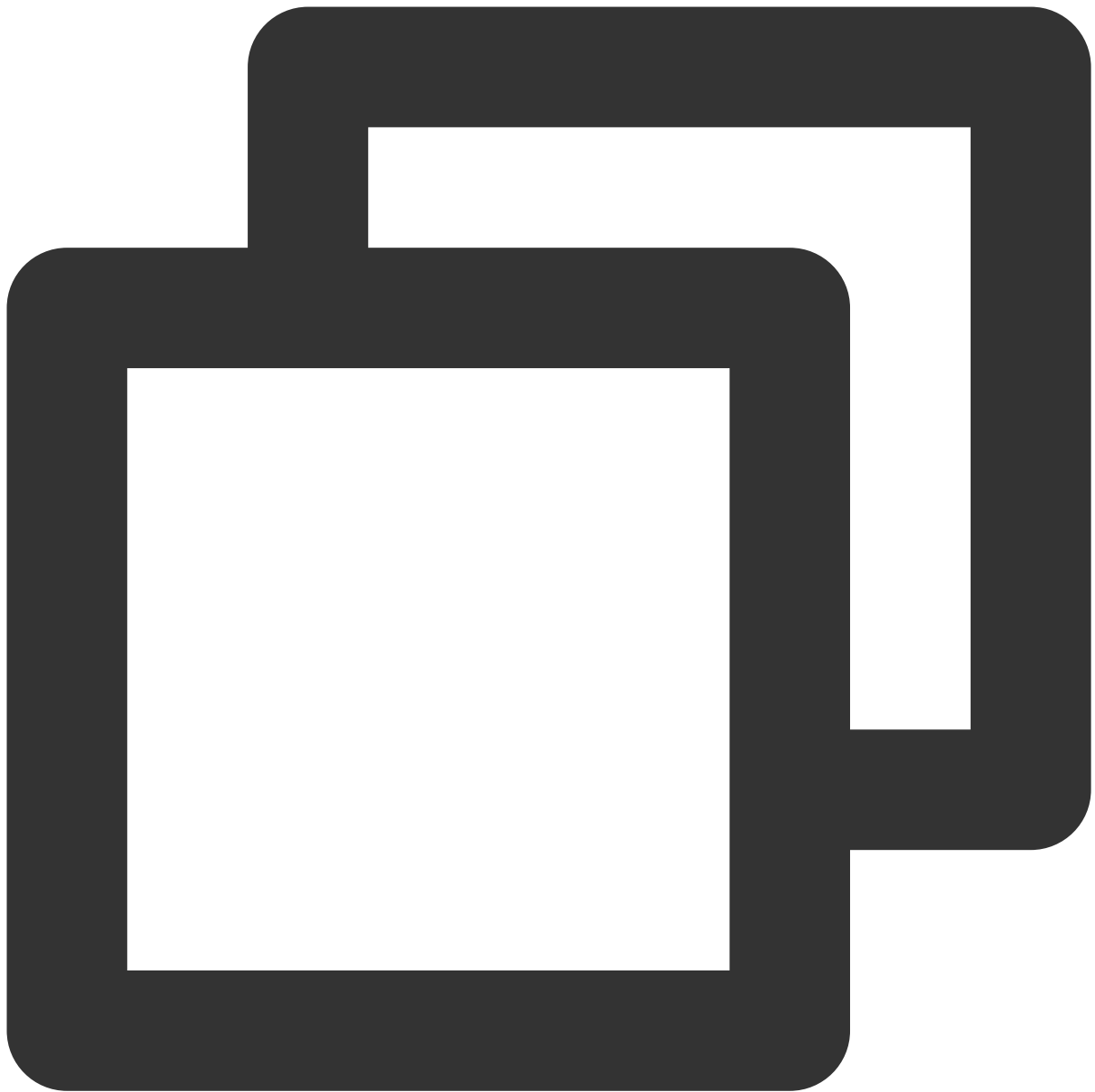
How do I view Cloud Contact Center logs?

The logs of Cloud Contact Center are compressed and encrypted by default, with suffix .log.

iOS log path: `sandbox/Documents/tccc`

Are the callbacks on iOS all on the main thread?

All callbacks in the Swift and OC interfaces are on the main thread, so developers do not need to handle them specially. However, callbacks in c++ are not on the main thread and need to be assessed by the business layer and then switched to the main thread:



```
if ([NSThread isMainThread]) {  
    // On the main thread, you can directly process  
    return;  
}  
dispatch_async(dispatch_get_main_queue(), ^{  
    // Callbacks are made from a non-main thread.  
});
```

Agent SDK APIs

Web

Last updated : 2024-04-01 18:07:37

Note

TCCC is the global variable accessed directly after the SDK is loaded.

Common Structure

AgentStatus

Agent status.

Field	Description
free	Idle
busy	In line
arrange	After-call-work
notReady	Busy
rest	On break

ServerType

Endpoint service type, the endpoint type used for describing phone sessions.

Field	Description
staffSeat	Web agent type
staffPhoneSeat	Agent's mobile type
miniProgramSeat	Mini program type
staffExtensionSeat	Telephone type

CommonSDKResponse

Parameter	Type	Required	Remarks

options	status	'success' 'error'	Yes	Result of SDK API call. Returns 'success' on success, and 'error' on failure.
	errorMsg	string	No	Error message, returned when status is 'error'

Call (API Functions of Telephone Customer Service and Audio Customer Service)

Inbound call

tccc.Call.startOutboundCall(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	phoneNumber	String	Yes	Contact Number
	phoneDesc	String	No	Number remarks, which will replace the number displayed in the call bar
	uui	String	No	User-defined data, which can be returned by pushing through Telephone CDR data push after input
	skillGroupId	String	No	The bound outbound number in the skill group
	callerPhoneNumber	String	No	Outbound number
	servingNumberGroupIds	String[]	No	Number ID list
	phoneEncodeType	'number'	No	Currently, only 'number' is supported, forcing the use of actual numbers when number mapping is enabled.

tccc.Call.startOutboundCall(options): Promise<CallResponse>

The description of CallResponse is as follows:

Parameter		Type	Required	Remarks
response	sessionId	String	Yes	Session ID
	calleeLocation	String	No	Called number's location
	calleePhoneNumber	String	Yes	Contact Number

	callerPhoneNumber	String	Yes	The calling number used for making an outbound call
	serverType	String	Yes	Type of the endpoint used for making an outbound call. Optional values are: staffSeat, staffPhoneSeat, and staffExtensionSeat. For details, see Session Service Type .
	remark	String	No	Called number remarks

Answers a call.

tccc.Call.accept(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID, obtained from the tccc.events.callIn event

Ends a conversation.

tccc.Call.hungUp(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Deleting a Session

tccc.Call.deleteCall(options)

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Mutes.

tccc.Call.muteMic(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Unmutes.

tccc.Call.unmuteMic(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Is It Currently Muted**tccc.Call.isMicMuted(options): Promise<CommonSDKResponse>**

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Initiating an Internal Call**tccc.Call.startInternalCall(): Promise<CommonSDKResponse>**

Parameter		Type	Required	Remarks
options	calleeUserId	String	Yes	Called agent account
	useMobile	Boolean	No	Whether to call their mobile phone or not

Transferring a Session**tccc.Call.transfer(): Promise<CommonSDKResponse>**

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	skillGroupId	String	No	Transfers to a specified skill group.
	userId	String	No	Transfers to a specified agent.

Call hold**tccc.Call.hold(): Promise<CommonSDKResponse>**

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Canceling Call Hold

tccc.Call.unHold(): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Sending Extension Number

tccc.Call.sendDigits(): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	dtmfText	String	No	The extension number to be sent

Chat (Online Customer Service API Functions)

Answers a call.

tccc.Chat.accept(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Ending a Session

tccc.Chat.end(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Transferring a Session

tccc.Chat.transfer(): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks

options	sessionId	String	Yes	Session ID
	skillGroupId	String	No	Transfers to a specified skill group.
	userId	String	No	Transfers to a specified agent.

Video (Video Customer Service API Functions)

Answers a call.

tccc.Video.accept(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Ends a conversation.

tccc.Video.end(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Mutes.

tccc.Video.muteMic(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Unmutes.

tccc.Video.unmuteMic(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Disabling the Camera

tccc.Video.muteVideo(options): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Enabling the Camera**tccc.Video.unmuteVideo(options): Promise<CommonSDKResponse>**

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Transferring a Session**tccc.Video.transfer(): Promise<CommonSDKResponse>**

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	skillGroupId	String	No	Transfers to a specified skill group.
	userId	String	No	Transfers to a specified agent.

Agent (Agent Status API Functions)

For more agent status enumeration types, refer to [Agent Status](#).

Going Online

tccc.Agent.online(): void

Going Offline

tccc.Agent.offline(): void

Setting Agent Status**tccc.Agent.setStatus(opts): Promise<CommonSDKResponse>**

Parameter		Type	Required	Remarks

options	status	String	Yes	Agent status, valid values: free: Idle rest: On break arrange: After-call-work notReady: Busy stopNotReady: Stops showing busy
	restReason	String	No	On break reason

Obtaining Agent Status

`tccc.Agent.getStatus():AgentStatus`

Devices (Device API Functions)

Checking Whether the Browser Is Supported

`tccc.Devices.isBrowserSupported(): boolean`

Note

TCCC Web SDK supports Google Chrome 56 and Microsoft Edge 80 or later.

Returning to Microphone List

`tccc.Devices.getMicrophones(): Promise<MediaDeviceInfo []>`

Returning to Speaker List

`tccc.Devices.getSpeakers(): Promise<MediaDeviceInfo []>`

UI (User Interface API Functions)

Hiding All SDK UIs

`tccc.UI.hide(): void`

Showing All SDK UIs

`tccc.UI.show(): void`

Showing Floating Button

`tccc.UI.showfloatButton(): void`

Hiding Floating Button

`tccc.UI.hidefloatButton(): void`

Showing Workspace

`tccc.UI.showWorkbench(): void`

Hiding Workspace

`tccc.UI.hideWorkbench(): void`

Events

Listening to Events

`tccc.on(event, callback)`

Canceling Event Listening

`tccc.off(event, callback)`

Completing SDK Initialization

`tccc.events.ready`

Triggered when SDK initialization is complete, at which point the API can be called safely.

Incoming Session

`tccc.events.callIn`

Types of incoming sessions include:

phone: Telephone session

im: Online session

voip: Audio session

video: Video session

internal: Internal session

Incoming Telephone Session

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	type	'phone'	Yes	Telephone session type

timeout	Number	Yes	Session access timeout duration, with 0 representing no timeout
calleePhoneNumber	String	Yes	Contact Number
callerPhoneNumber	String	No	Caller Number
callerLocation	String	No	Caller's location
remark	String	No	Remarks
ivrPath	{key: String, label: String}[]	-	User's IVR key path. key represents the corresponding key, and label represents the corresponding key tag.
protectedCallee	String	No	When mapping is enabled, the number represents the called party.
protectedCaller	String	No	When mapping is enabled, the number represents the caller.
serverType	'staffSeat' 'staffPhoneSeat' 'staffExtensionSeat'	Yes	Indicates the type of the agent the call goes to. staffSeat: Default value, indicating Web agent StaffPhoneSeat: Agent's mobile MiniProgramSeat: Mini program agent staffExtensionSeat: Telephone bound to the agent

Incoming Online Session

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	type	'phone'	Yes	Telephone session type
	timeout	Number	Yes	Session access timeout duration, with 0 representing no timeout
	nickname	String	Yes	User's nickname
	avatar	String	No	User's profile photo
	remark	String	No	Remarks
	peerSource	String	No	Channel source

	channelName	String	No	Custom parameter
	clientData	String	No	User-defined parameter

Incoming Audio Session

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	type	'voip'	Yes	Audio session type
	timeout	Number	Yes	Session access timeout duration, with 0 representing no timeout
	callee	String	Yes	Channel entry
	calleeRemark	String	No	Channel entry remarks
	userId	String	Yes	User's openId
	nickname	String	No	Users will get a WeChat nickname after authorization.
	avatar	String	No	Users will receive a WeChat profile photo after authorization.
	remark	String	No	Remarks
	peerSource	String	No	Caller's location
	ivrPath	{key: String, label: String}[]	No	User's IVR key path. key represents the corresponding key, and label represents the corresponding key tag.
	clientData	String	No	User-defined parameter

Incoming Video Session

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	type	'video'	Yes	Video session type
	timeout	String	Yes	Session access timeout duration, with 0 representing no timeout

	userId	String	Yes	User's openId
	nickname	String	No	Users will get a WeChat nickname after authorization.
	avatar	String	No	Users will receive a WeChat profile photo after authorization.
	remark	String	No	Remarks

Incoming Internal Session

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	type	'internal'	Yes	Internal session type
	timeout	Number	Yes	Session access timeout duration, with 0 representing no timeout
	peerUserId	String	Yes	Caller's account

Agent Access Session

tccc.events.userAccessed

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Session Timeout Transfer Event

tccc.events.autoTransfer

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Session End Event

tccc.events.sessionEnded

Parameter	Type	Required	Remarks
-----------	------	----------	---------

options	sessionId	String	Yes	Session ID
	closeBy	String	Yes	Indicates the hang-up party: client: Hang up by user seat: Hang up by agent admin: Hang up by system timer: Hang up by timer
	mainReason	String	No	Only exists in telephone type, and when the hang-up party is "admin", indicating the reason for hang-up.
	subReason	String	No	Only exists in telephone type, and when the hang-up party is "admin", indicating the detailed reason for hang-up.

Successful Outbound Call Event

tccc.events.callOuted

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Outbound Call Answering Event

tccc.events.calloutAccepted

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Session Transfer Event

tccc.events.transfer

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Agent Status Change Event

tccc.events.statusChanged

Parameter		Type	Required	Remarks	
options	status	AgentStatus	No	For details, refer to Agent Status .	

Automatic Speech Recognition Event

tccc.events.asr

Parameter		Type	Required	Remarks	
options	sessionId	String	Yes	Session ID	
	result	ASR recognition result	Yes	Automatic Speech Recognition result structure	
	flow	'IN' 'OUT'	Yes	Recognition direction IN: User side OUT: Agent side	

uni-app

Last updated : 2024-04-01 17:50:44

API Overview

Creating Instances and Event Callbacks

API	Description
sharedInstance	Creates a TCCCWorkstation instance (singleton).
destroyInstance	Destroys a TCCCWorkstation instance (singleton). It is recommended to uninstall the TCCCWorkstation instance when it is not in use.
on	Sets a TCCCWorkstation event callback.
off	Cancel s a TCCCWorkstation event callback.

Sample Code for Creating Instances and Setting Event Callbacks



```
// Import TCCC-related package
import {TcccWorkstation,TCCCLoginType,TCCCAudioRoute,TCCCEndReason} from "tccc-sdk-
// Create an instance and set an event callback
const tcccSDK = TCCCWorkstation.sharedInstance();
// Error event callback
tcccSDK.on('onError',(errCode,errMsg) => {
});
// Call end callback
tcccSDK.on('onEnded',(reason,reasonMessage,sessionId) => {
  if (reason == TCCCEndReason.Error) {
    // Call exception
```

```
    }  
  });  
  // Peer answer callback  
  tcccSDK.on('onAccepted', (sessionId) => {  
  
  });  
  // Release all event callback monitoring  
  tcccSDK.off('*');
```

Login API Functions

API	Description
login	SDK login
checkLogin	Checks SDK login status. It is recommended to call when the page onShow .
logout	SDK logout

Login Sample Code



```
// For how to obtain sdkAppId, userId, and token, see the corresponding fields in K
// Agent login
tcccSDK.login({
  sdkAppID: sdkAppId,
  userId: userID,
  token: token,
  type: type,
}, (code,message) => {
  if (code == TcccErrorCode.ERR_NONE) {
    // Login succeeded
  } else {
```

```
        // Login failed
    }
});
// When the mobile application is switched to the background, the operating system
tcccSDK.checkLogin(code,message) => {
    if (code == TcccErrorCode.ERR_NONE) {
        // Logged in
    } else {
        // Not logged in
    }
}
});
```

Call-related API Functions

API	Description
call	Initiates a call
answer	Answers the inbound call
terminate	Ends the call
sendDTMF	Sends Dual-Tone Multi-Frequency (DTMF) signals
mute	Mutes.
unmute	Unmutes.
startPlayMusic	Starts playing music
stopPlayMusic	Stops playing music

Sample Code for Initiating and Ending a Call



```
// Initiate a call
tcccSDK.call({
  to: '134xxxx',           // Contact number (required)
  remark: "xxx",           // Number remarks, which will replace the number displayed
  uui: "xxxx",             // User-defined data (optional)
}, (code,message) => {
  if (code == TcccErrorCode.ERR_NONE) {
    // Initiation succeeded
  } else {
    // Initiation failed
  }
}
```

```
});  
  
// End the call  
tcccSDK.terminate();  
// Answer the call  
tcccSDK.answer((code,message) => {  
    if (code == TcccErrorCode.ERR_NONE) {  
        // Answer succeeded  
    } else {  
        // Answer failed  
    }  
});
```

Audio Device API Functions

API	Description
setAudioCaptureVolume	Sets the local audio capture volume.
getAudioCaptureVolume	Obtains the local audio capture volume.
setAudioPlayoutVolume	Sets the remote audio playback volume.
getAudioPlayoutVolume	Obtains the remote audio playback volume.
setAudioRoute	Sets audio routing.



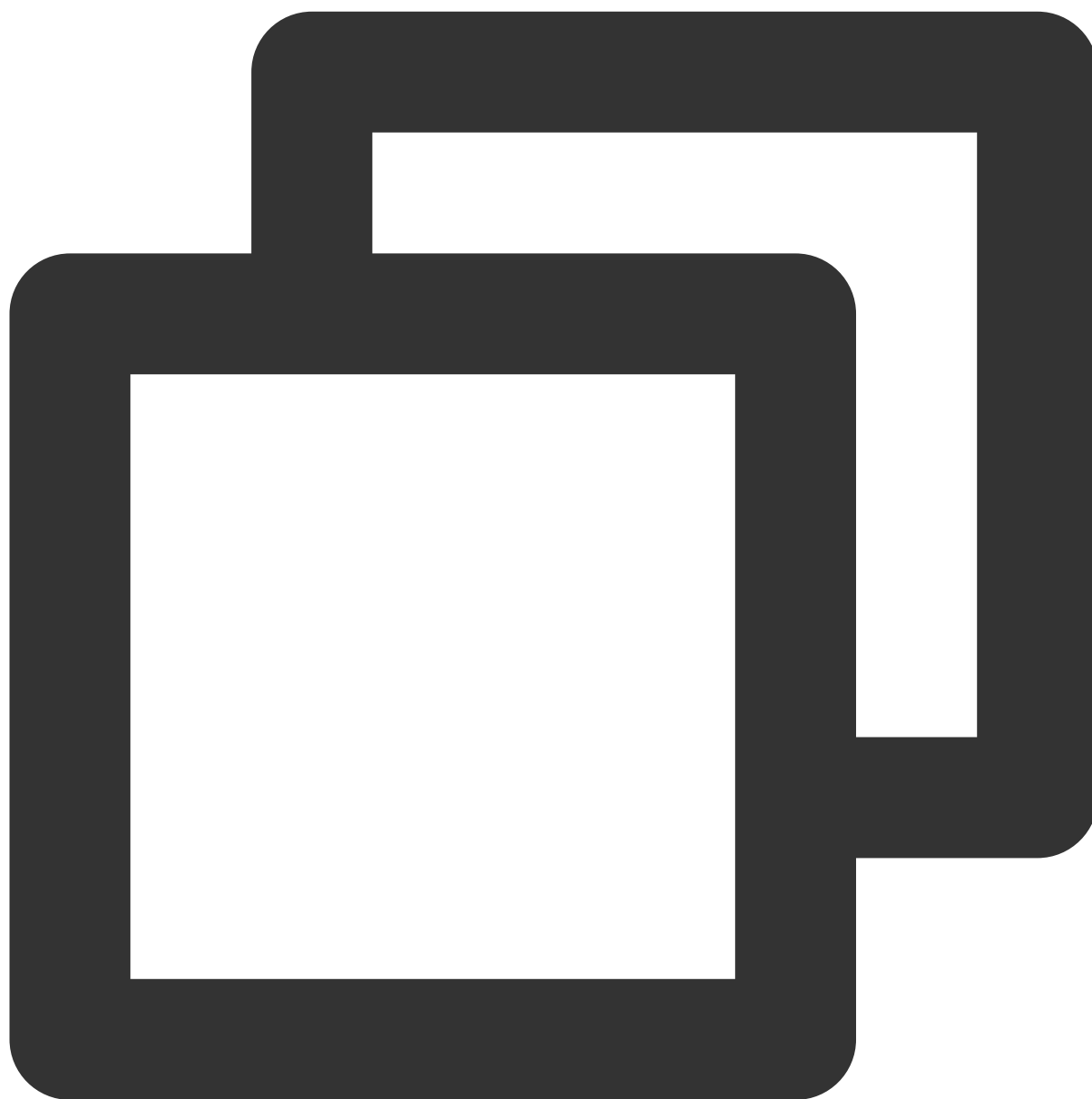
```
// TCCCAudioRoute.Earpiece is an earphone
// Set as a speaker
const route = TCCCAudioRoute.Speakerphone;
tcccSDK.getDeviceManager().setAudioRoute(route);
```

Debugging APIs

API	Description
getSDKVersion	Obtains the SDK version.

setLogLevel	Sets the log output level.
setConsoleEnabled	Enables/Disables console log print.

Sample Code for Obtaining SDK Version

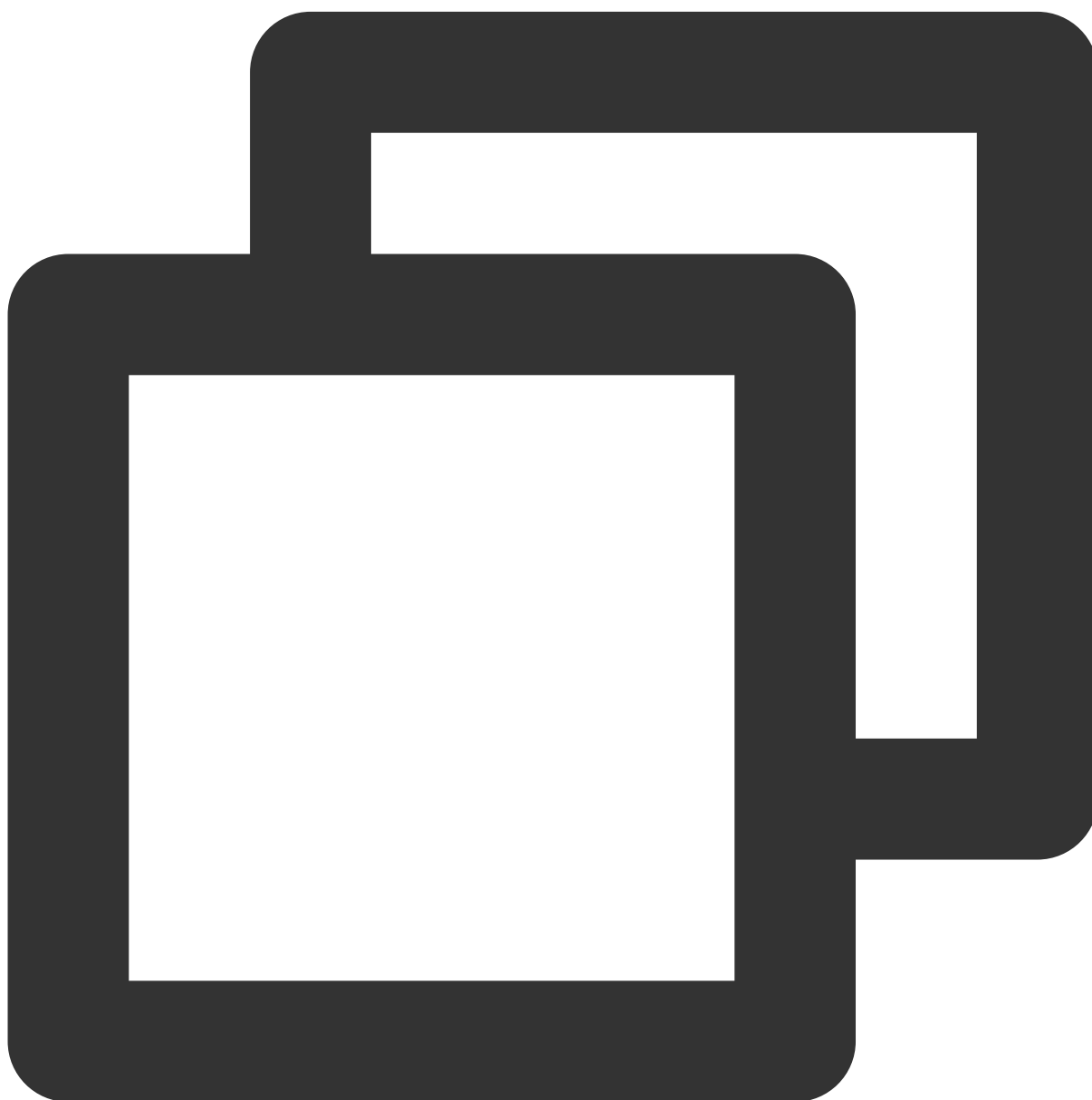


```
// Obtain SDK version number  
TCCCWorkstation.getSDKVersion();
```

Error and Warning Events

API	Description
onError	Error event callback
onWarning	Warning event callback

Sample Code for Handling Error Event Callbacks



```
// Error event callback
```

```
tcccSDK.on('onError', (errCode, errMsg) => {
});
// Warning event callback
tcccSDK.on('onWarning', (warningCode, warningMsg) => {
});
```

Call Event Callback

API	Description
onNewSession	New session event, including inbound and outbound calls
onAccepted	Peer answer callback
onEnded	Session End Event
onAudioVolume	Callback for volume feedback
onNetworkQuality	Real-time statistics callback of network quality

Sample Code for Answer and Agent Hang-up Event Callbacks



```
// Session end event
tcccSDK.on("onEnded", (reason, reasonMessage, sessionId) => {
  var msg = reasonMessage;
  if (reason == TCCCEndReason.Error) {
    msg = "System exception "+reasonMessage;
  } else if (reason == TCCCEndReason.Timeout) {
    msg = "Timeout hang-up";
  } else if (reason == TCCCEndReason.LocalBye) {
    msg = "You hung up";
  } else if (reason == TCCCEndReason.RemoteBye) {
    msg = "The other party has hung up";
  }
});
```

```

    } else if (reason == TCCCEndReason.Rejected) {
        msg = "The other party has rejected";
    } else if (reason == TCCCEndReason.RemoteCancel) {
        msg = "The other party has cancelled";
    }
});
// New session event, including inbound and outbound calls
tcccSDK.on('onNewSession', (res) => {
    const sessionDirection = res.sessionDirection;
    if (sessionDirection == TCCCSessionDirection.CallIn) {
        // Inbound call. You cannot receive this event when the phone switches to t
    } else if (sessionDirection == TCCCSessionDirection.CallOut){
        // Outbound call
    }
});
// The other party has answered
tcccSDK.on('onAccepted', (sessionId) => {
});
// Real-time statistics callback of network quality
tcccSDK.on('onNetworkQuality', (localQuality) => {
    const quality = localQuality.quality;
    // Current network is average
    //   TCCCQuality_Poor = 3,
    // Current network is poor
    //   TCCCQuality_Bad = 4,
    // Current network is very poor
    //   TCCCQuality_Vbad = 5,
    // The current network does not meet the minimum requirements for calls
    //   TCCCQuality_Down = 6,

});
// Callback for volume feedback. Volume is from 0 to 100, and a larger value indica
tcccSDK.on('onAudioVolume', (userId, volume) => {

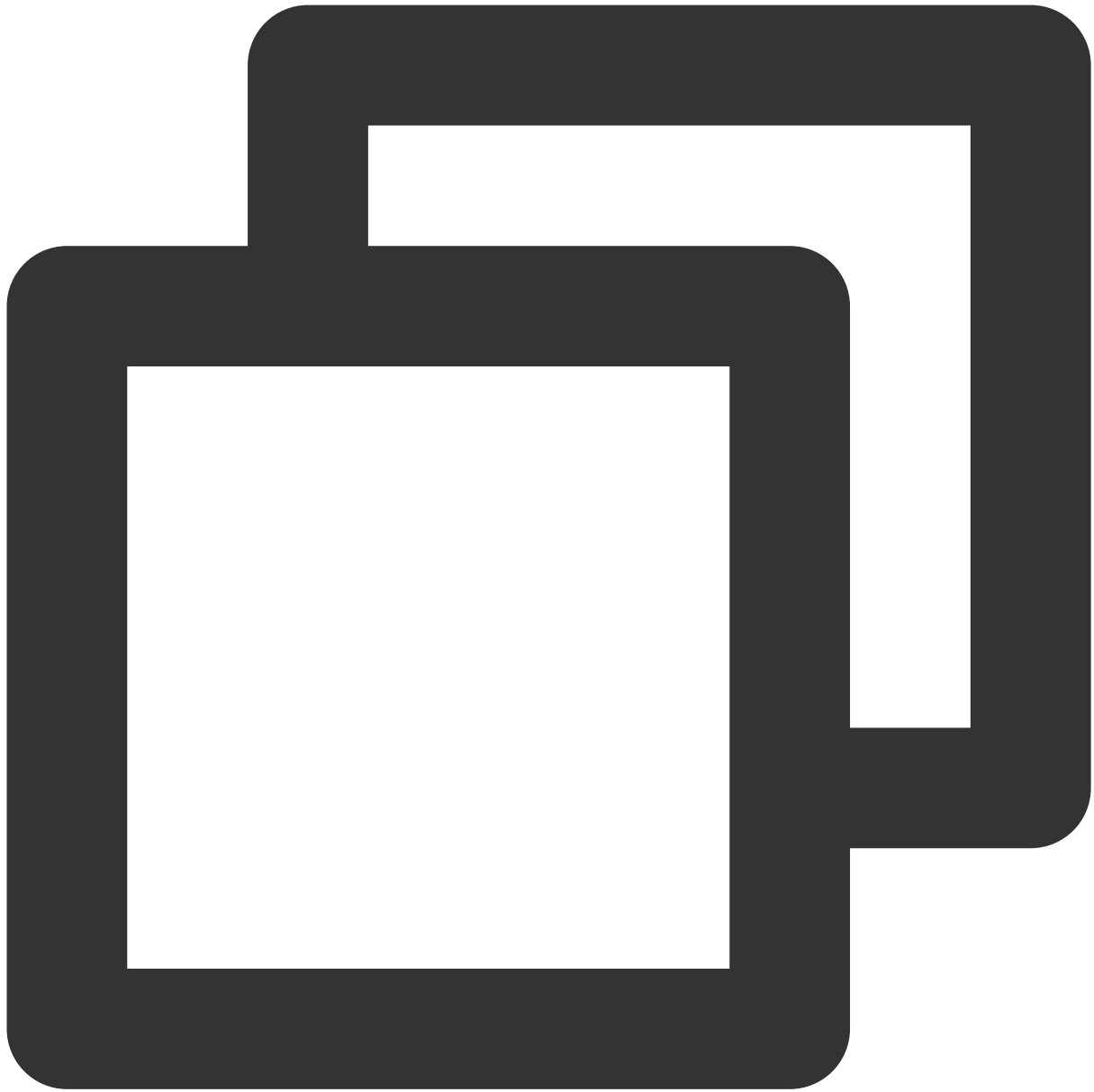
});

```

Event Callback of Connection with Cloud

API	Description
onConnectionLost	The connection between the SDK and the cloud has been disconnected.
onTryToReconnect	The SDK is trying to reconnect to the cloud.
onConnectionRecovery	The connection between the SDK and the cloud has been restored.

Sample Code for Event Callback of Connection with Cloud



```
tcccSDK.on('onConnectionLost', (serverType) => {  
    // The connection with the cloud has been disconnected  
});  
tcccSDK.on('onTryToReconnect', (serverType) => {  
    // Trying to reconnect to the cloud  
});  
tcccSDK.on('onConnectionRecovery', (serverType) => {  
    // The connection with the cloud has been restored  
});
```

API Error Codes

Basic Error Codes

Symbol	Value	Meaning
ERR_NONE	0	No error. Succeeded.
ERR_HTTP_REQUEST_FAILURE	-10001	HTTP request failed. Please check your network connection.
ERR_HTTP_TOKEN_ERROR	-10002	The token login ticket is incorrect or has expired.
ERR_HTTP_GETSIPINFO_ERROR	-10003	Failed to obtain the agent configuration. Please contact us.
ERR_NETWORK_CANNOT_RESET	-10004	In a call. Network reset and outbound call are prohibited.
ERR_HAD_LOGGEDOUT	-10005	You have already logged out. Please log in again.
ERR_UNRIGIST_FAILURE	20001	Failed to deregister.
ERR_ANSWER_FAILURE	20002	Failed to answer the call, usually because the TRTC failed to enter the room.
ERR_SIPURI_WRONGFORMAT	20003	URI format error.

SIP Error Codes

Symbol	Value	Meaning
ERR_SIP_BAD_REQUEST	400	Error request, usually because the agent initiates a request without logging in
ERR_SIP_UNAUTHORIZED	401	Unauthorized (username or password is incorrect)
ERR_SIP_PAYMENTREQUIRED	402	Payment required, typically when the agent's license is full
ERR_SIP_FORBIDDEN	403	Incorrect password, or has been kicked out
ERR_SIP_REQUESTTIMEOUT	408	Request timeout (network timeout)
ERR_SIP_REQUEST_TERMINATED	487	Request termination (network error, in case of network

		interruption)
ERR_SIP_SERVICE_UNAVAILABLE	503	Service unavailable
ERR_SIP_SERVER_TIMEOUT	504	Service timeout

Audio Device Error Codes

Symbol	Value	Meaning
ERR_MIC_START_FAIL	-1302	Failed to start the microphone. The device's microphone configuration program (driver) is abnormal. Please disable and re-enable the device, restart the device, or update the configuration program.
ERR_MIC_NOT_AUTHORIZED	-1317	No access to the microphone. This usually occurs on mobile devices and may be because the user denied the access.
ERR_MIC_SET_PARAM_FAIL	-1318	Failed to set microphone parameters.
ERR_MIC_OCCUPY	-1319	The microphone is occupied. This occurs when, for example, the user is currently having a call on the mobile device.
ERR_MIC_STOP_FAIL	-1320	Failed to stop the microphone.
ERR_SPEAKER_START_FAIL	-1321	Failed to start the speaker, for example, on Windows or Mac.
ERR_SPEAKER_SET_PARAM_FAIL	-1322	Failed to set speaker parameters.
ERR_SPEAKER_STOP_FAIL	-1323	Failed to stop the speaker.
ERR_UNSUPPORTED_SAMPLERATE	-1306	Unsupported audio sample rate

Network Error Codes

Symbol	Value	Meaning
ERR_RTC_ENTER_ROOM_FAILED	-3301	Failed to enter the room. Please view -3301 in onError to confirm the message for the reason of the failure.
ERR_RTC_REQUEST_IP_TIMEOUT	-3307	Request for IP and Sig timed out. Please check whether your network is functioning properly and

		whether UDP is unblocked in your network firewall.
ERR_RTC_CONNECT_SERVER_TIMEOUT	-3308	Request for room entry timed out. Please check your network connection or whether you are on a VPN. You can also try switching to 4G for confirmation.
ERR_RTC_ENTER_ROOM_REFUSED	-3340	Room entry request was denied. Please check whether you are continually calling enterRoom to enter the room of the same ID.

Android

Last updated : 2024-04-01 17:51:12

Creating Instances and Event Callbacks

API	Description
sharedInstance	Creates a TCCCWorkstation instance (singleton).
destroySharedInstance	Destroys a TCCCWorkstation instance (singleton).
setListener	Sets a TCCCWorkstation event callback.

Sample Code for Creating Instances and Setting Event Callbacks



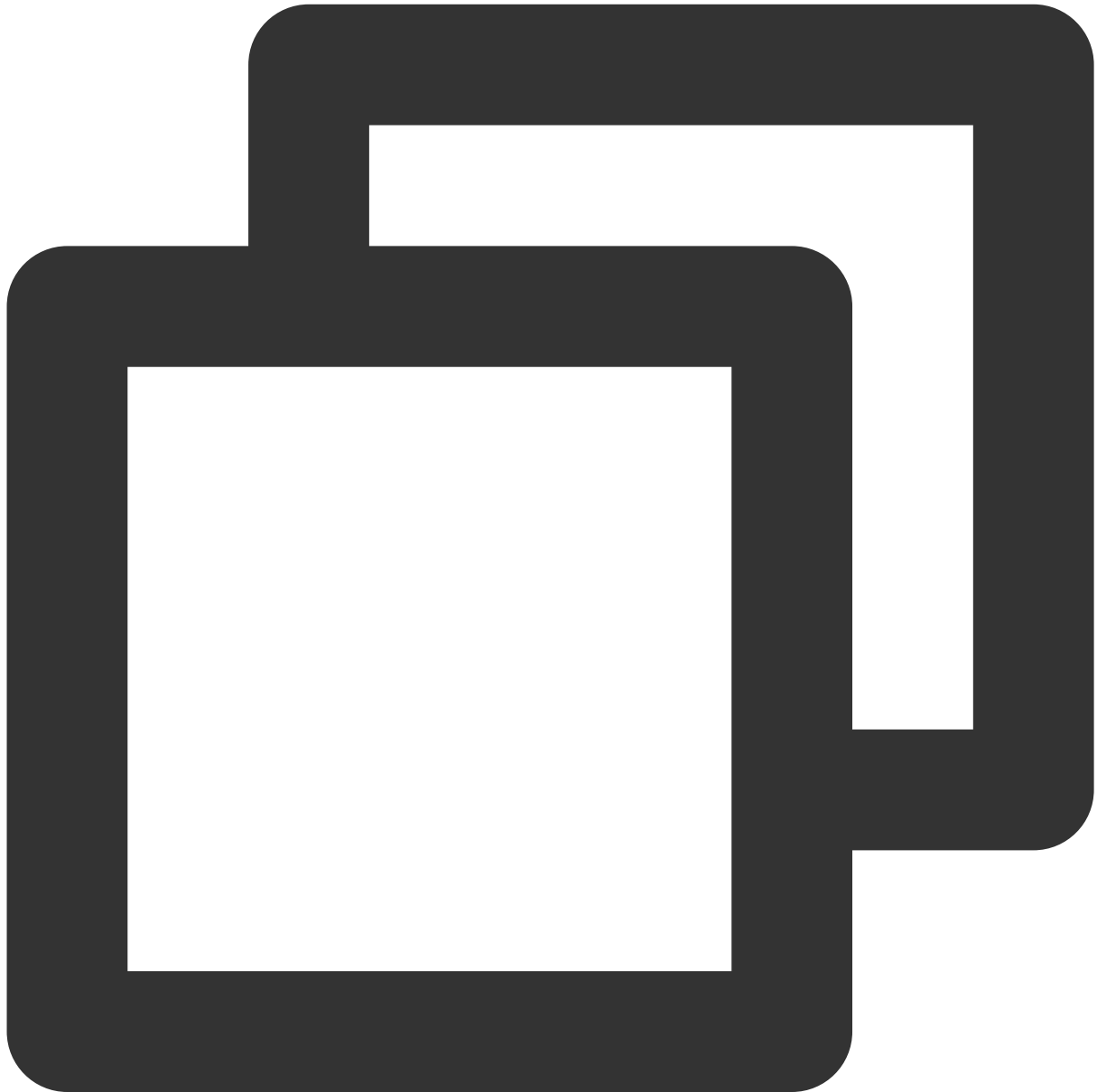
```
// Create an instance and set an event callback
TCCCWorkstation tcccSDK = TCCCWorkstation.sharedInstance(getApplicationContext());
tcccSDK.setListener(new TCCCListener() {});
```

Login API Functions

API	Description
login	SDK login

<code>checkLogin</code>	Checks whether the SDK is already logged in.
<code>logout</code>	SDK logout

Login Sample Code



```
TCCCTypeDef.TCCCLoginParams loginParams = new TCCCTypeDef.TCCCLoginParams();  
/// The agent ID for login, which is usually an email address  
loginParams.userId = "";  
/// The login ticket, required for the Agent login mode. For details, see Creating  
/// Token] (https://cloud.tencent.com/document/product/679/49227)
```

```
loginParams.token = "";  
/// Cloud Contact Center application ID, which usually starts with 1400  
loginParams.sdkAppId = 0;  
// Must be the Agent mode  
loginParams.type = TCCCTypeDef.TCCCLoginType.Agent;  
  
tcccSDK.login(loginParams, new TXCallback() {  
    @Override  
    public void onSuccess() {  
        // login success  
    }  
  
    @Override  
    public void onError(int code, String desc) {  
        // login error  
    }  
});
```

Call-related API Functions

API	Description
call	Initiates a call
answer	Answers the inbound call
terminate	Ends the call
sendDTMF	Sends Dual-Tone Multi-Frequency (DTMF) signals
mute	Mutes.
unmute	Unmutes.

Sample Code for Initiating and Ending a Call



```
TCCCTypeDef.TCCCStartCallParams callParams =new TCCCTypeDef.TCCCStartCallParams();
//Format <scheme> : <user> @<host>, such as sip:1343xxxx@1400xxxx.tccc.qcloud.com,
callParams.to = "sip:1343xxxx@1400xxxx.tccc.qcloud.com";
// Initiate a call
tcccSDK.call(callParams, new TXCallback() {
    @Override
    public void onSuccess() {
        // call success
    }

    @Override
```

```
public void onError(int code, String desc) {  
    // call error  
}  
});  
// End the call  
tcccSDK.terminate();
```

Audio Device API Functions

API	Description
setAudioCaptureVolume	Sets the local audio capture volume.
getAudioCaptureVolume	Obtains the local audio capture volume.
setAudioPlayoutVolume	Sets the remote audio playback volume.
getAudioPlayoutVolume	Obtains the remote audio playback volume.
setAudioRoute	Sets audio routing.

Debugging APIs

API	Description
getSDKVersion	Obtains the SDK version.
setLogLevel	Sets the log output level.
setConsoleEnabled	Enables/Disables console log print.
callExperimentalAPI	Calls an experimental API.

Sample Code for Obtaining SDK Version

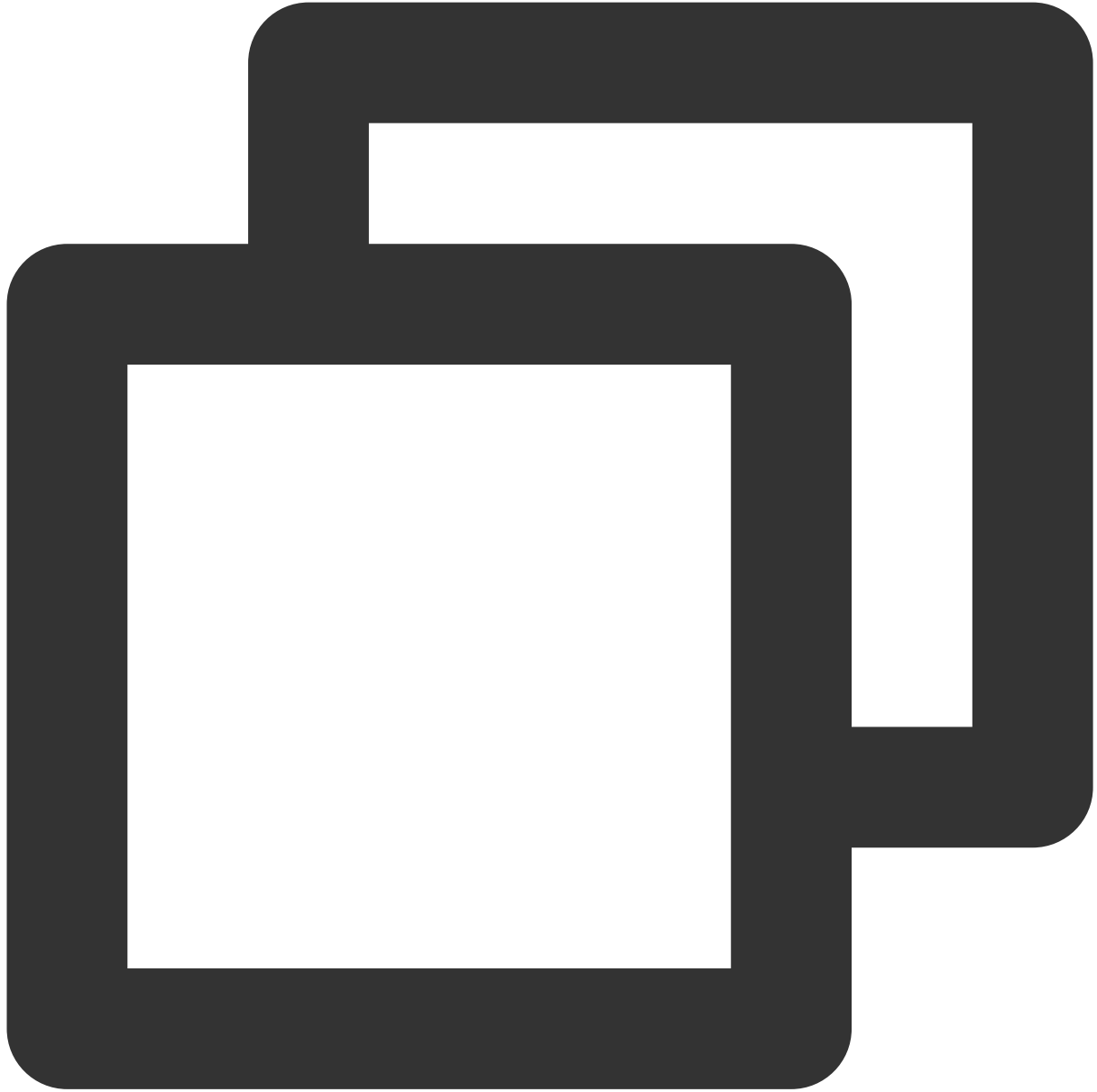


```
// Obtain SDK version number  
TCCCWorkstation.getSDKVersion();
```

Error and Warning Events

API	Description
onError	Error event callback
onWarning	Warning event callback

Sample Code for Handling Error Event Callbacks



```
tcccSDK.setListener(new TCCCListener() {  
    /**  
     * Error event callback  
     * Error event, indicating that the SDK encounters an irrecoverable error, s  
     * @param errCode    //Error code  
     * @param errMsg     //Error message  
     * @param extraInfo  Additional information field. Some error codes may carry  
     */  
    @Override
```

```
public void onError(int errCode, String errMsg, Bundle extraInfo) {
    super.onError(errCode, errMsg, extraInfo);
}

/**
 * Warning event callback
 * Warning event, indicating advisory issues thrown by the SDK, such as audi
 * @param warningCode Warning code
 * @param warningMsg Warning message
 * @param extraInfo Additional information field. Some warning codes may c
 */
@Override
public void onWarning(int warningCode, String warningMsg, Bundle extraInfo) {
    super.onWarning(warningCode, warningMsg, extraInfo);
}
});
```

Call Event Callback

API	Description
onNewSession	New session event, including inbound and outbound calls
onEnded	Session End Event
onAudioVolume	Callback for volume feedback
onNetworkQuality	Real-time statistics callback of network quality

Sample Code for Answer and Agent Hang-up Event Callbacks



```
tcccSDK.setListener(new TCCCListener() {  
    @Override  
    public void onNewSession(TCCCTypeDef.ITCCCSessionInfo info) {  
        super.onNewSession(info);  
        // New session event, including inbound and outbound calls. The direction can  
    }  
  
    @Override  
    public void onEnded(int reason, String reasonMessage, String sessionId) {  
        super.onEnded(reason, reasonMessage, sessionId);  
        // End of session  
    }  
});
```

```
}

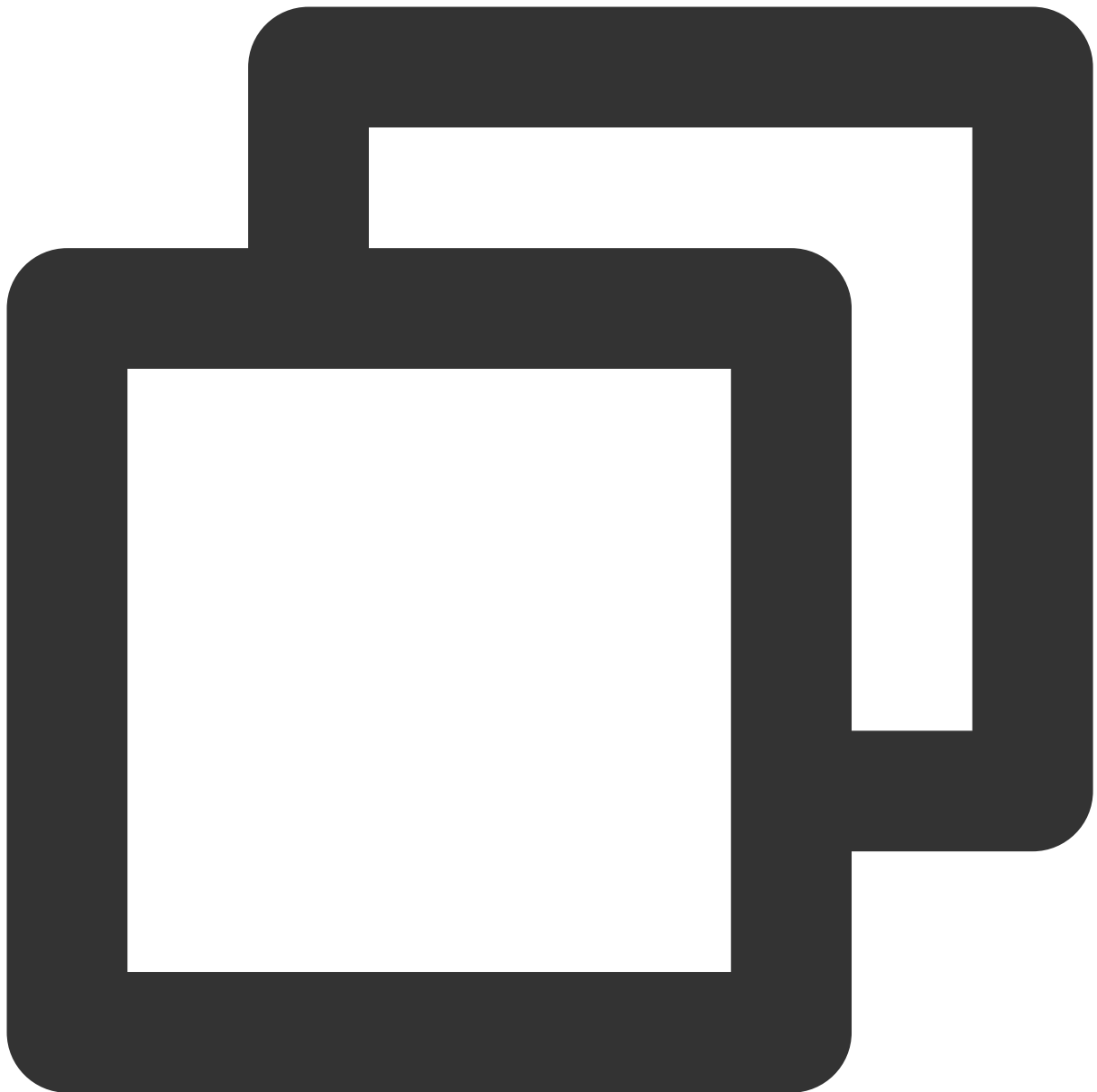
@Override
public void onAccepted(String sessionId) {
    super.onAccepted(sessionId);
    // Counterpart answers
}

});
```

Event Callback of Connection with Cloud

API	Description
onConnectionLost	The connection between the SDK and the cloud has been disconnected.
onTryToReconnect	The SDK is trying to reconnect to the cloud.
onConnectionRecovery	The connection between the SDK and the cloud has been restored.

Sample Code for Event Callback of Connection with Cloud



```
tcccSDK.setListener(new TCCCListener() {  
    /**  
     * The connection between the SDK and the cloud has been disconnected.  
     * The SDK will issue this callback when its connection with the cloud is di  
     * For example, this event may occur when the user enters an elevator during  
     * During the reconnection process, onTryToReconnect will be thrown, and onC  
     * Therefore, the SDK switches between the following three connection-relate  
     */  
    @Override  
    public void onConnectionLost(TCCCServerType serverType) {  
        super.onConnectionLost(serverType);  
    }  
});
```

```
}

/**
 * The SDK is trying to reconnect to the cloud
 * When the SDK's connection with the cloud is lost, it will throw onConnect
 * After the connection is restored, onConnectionRecovery is thrown.
 */
@Override
public void onTryToReconnect(TCCServerType serverType) {
    super.onTryToReconnect(serverType);
}

/**
 * The connection between the SDK and the cloud has been restored.
 * When the SDK's connection with the cloud is lost, it throws onConnectionL
 * This callback is thrown when the connection is restored.
 */
@Override
public void onConnectionRecovery(TCCServerType serverType) {
    super.onConnectionRecovery(serverType);
}
});
```

API Error Codes

Basic Error Codes

Symbol	Value	Meaning
ERR_SIP_SUCCESS	200	Execution succeeded.
ERR_UNRIGIST_FAILURE	20001	Login failed.
ERR_ANSWER_FAILURE	20002	Failed to answer the call, usually because the TRTC failed to enter the room.
ERR_SIPURI_WRONGFORMAT	20003	URI format error

SIP Error Codes

Symbol	Value	Meaning
ERR_SIP_BAD_REQUEST	400	Error request

ERR_SIP_UNAUTHORIZED	401	Unauthorized (username or password is incorrect)
ERR_SIP_AUTHENTICATION_REQUIRED	407	Proxy authentication required. Please check whether the login API has been called.
ERR_SIP_REQUESTTIMEOUT	408	Request timeout (network timeout)
ERR_SIP_REQUEST_TERMINATED	487	Request termination (network error, in case of network interruption)
ERR_SIP_SERVICE_UNAVAILABLE	503	Service unavailable
ERR_SIP_SERVER_TIMEOUT	504	Service timeout

Audio Device Error Codes

Symbol	Value	Meaning
ERR_MIC_START_FAIL	-1302	Failed to start the microphone. The device's microphone configuration program (driver) is abnormal. Please disable and re-enable the device, restart the device, or update the configuration program.
ERR_MIC_NOT_AUTHORIZED	-1317	No access to the microphone. This usually occurs on mobile devices and may be because the user denied the access.
ERR_MIC_SET_PARAM_FAIL	-1318	Failed to set microphone parameters.
ERR_MIC_OCCUPY	-1319	The microphone is occupied. This occurs when, for example, the user is currently having a call on the mobile device.
ERR_MIC_STOP_FAIL	-1320	Failed to stop the microphone.
ERR_SPEAKER_START_FAIL	-1321	Failed to start the speaker, for example, on Windows or Mac.
ERR_SPEAKER_SET_PARAM_FAIL	-1322	Failed to set speaker parameters.
ERR_SPEAKER_STOP_FAIL	-1323	Failed to stop the speaker.
ERR_UNSUPPORTED_SAMPLERATE	-1306	Unsupported audio sample rate

Network Error Codes

--	--	--

Symbol	Value	Meaning
ERR_RTC_ENTER_ROOM_FAILED	-3301	Failed to enter the room. Please view -3301 in onError to confirm the message for the reason of the failure.
ERR_RTC_REQUEST_IP_TIMEOUT	-3307	Request for IP and Sig timed out. Please check whether your network is functioning properly and whether UDP is unblocked in your network firewall.
ERR_RTC_CONNECT_SERVER_TIMEOUT	-3308	Request for room entry timed out. Please check your network connection or whether you are on a VPN. You can also try switching to 4G for confirmation.
ERR_RTC_ENTER_ROOM_REFUSED	-3340	Room entry request was denied. Please check whether you are continually calling enterRoom to enter the room of the same ID.

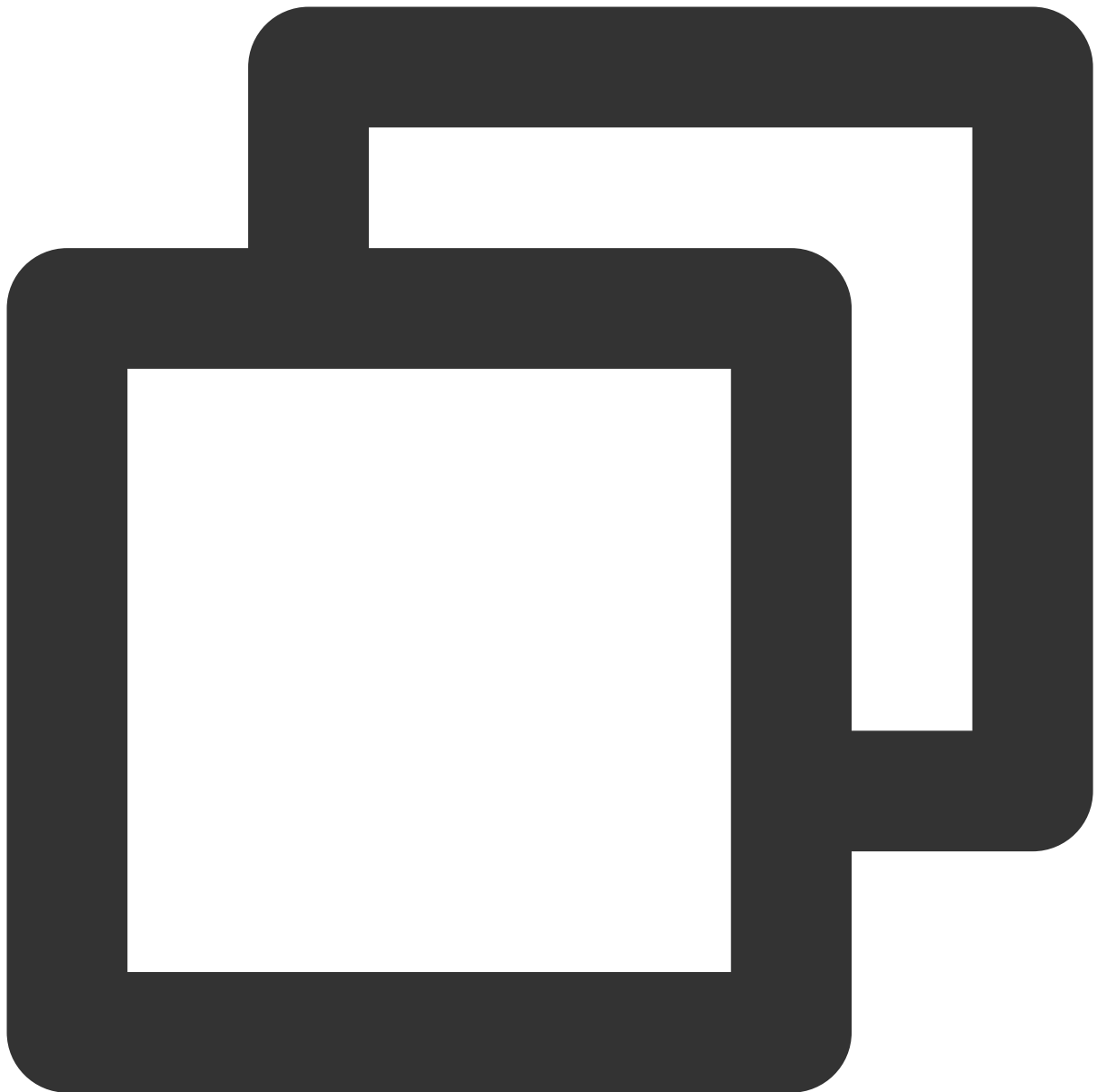
iOS

Last updated : 2024-04-01 17:52:16

Creating Instances and Event Callbacks

API	Description
getTCCCShareInstance	Creates an ITCCCWorkstation instance (singleton).
destroyTCCCShareInstance	Destroys an ITCCCWorkstation instance (singleton).
addCallback	Adds an ITCCCWorkstation event callback.
removeCallback	Removes an ITCCCWorkstation event callback.

Sample Code for Creating Instances and Setting Event Callbacks



```
#include "TCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;
// Create an instance and set an event callback
ITCCCWorkstation* tcccSDK = getTCCCShareInstance();
// Set the callback. TCCC_CALLBACK_IMPL needs to be derived from ITCCC_CALLBACK
class TCCC_CALLBACK_IMPL:public ITCCC_CALLBACK {
public:
    TCCC_CALLBACK_IMPL() {}
    ~TCCC_CALLBACK_IMPL() {}

    void onError(TCCError errCode, const char* errMsg, void* extraInfo) {}
};
```

```
void onWarning(TCCCCWarning warningCode, const char* warningMsg, void* extraInf

void onNewSession(TCCCSessionInfo info) {}

void onEnded(EndedReason reason, const char* reasonMessage, const char* session

};
TCCCCallbackImpl* tcccCallback = new TCCCCallbackImpl();
tcccSDK->addCallback(tcccCallback);
// Destroy the instance
destroyTCCCShareInstance();
tcccSDK = nullptr;
```

Login API Functions

API	Description
login	SDK login
checkLogin	Checks whether the SDK is already logged in.
logout	SDK logout

Login Sample Code



```
// Callback class for login
class TCCCLoginCallbackImpl : public ITXValueCallback<TCCCLoginInfo> {
public:
    TCCCLoginCallbackImpl() {

    }
    ~TCCCLoginCallbackImpl() override {}
    void OnSuccess(const TCCCLoginInfo &value) override {
        // Login succeeded
    }
}
```

```
void OnError(TCCCErrors error_code, const char *error_message) override {
    // Login failed
}

};
TCCCLoginCallbackImpl* loginCallbackImpl = nullptr;
if (nullptr == loginCallbackImpl) {
    loginCallbackImpl = new TCCCLoginCallbackImpl();
}
TCCCLoginParams param;
/// The agent ID for login, which is usually an email address
param.userId = "";
/// The login ticket, required for the Agent login mode. For details, see Creating
/// Token] (https://cloud.tencent.com/document/product/679/49227)
param.token = "";
/// Cloud Contact Center application ID, which usually starts with 1400
param.sdkAppId = 0;
// Must be the Agent mode
param.type = TCCCLoginType::Agent;
// Login
tcccSDK->login(param, loginCallbackImpl);
// Logout
tcccSDK->logout(nullptr);
```

Call-related API Functions

API	Description
call	Initiates a call.
answer	Answers a call.
terminate	Ends a call.
sendDTMF	Sends Dual Tone Multi-Frequency (DTMF) signals.
mute	Mutes.
unmute	Unmutes.

Sample Code for Initiating and Ending a Call



```
class TCCCCCommonCallback : public ITXCallback {
private:
    NSString* mFunName;
public:
    TCCCCCommonCallback(NSString* funName) {
        mFunName = funName;
    }
    ~TCCCCCommonCallback() override {

    }
    void OnSuccess() override {
```

```
// Succeeded
}

void OnError(TCCCErrors error_code, const char *error_message) override {
    std::string copyErrMsg = makeString(error_message);
    // Failed
}

};
TCCCommonCallback* startCallCallbackImpl = nullptr;
if (nullptr == startCallCallbackImpl) {
    startCallCallbackImpl = new TCCCommonCallback(@"startCall");
}
TCCStartCallParams callParams;
// Phone number for the call
callParams.to = "";
// Initiate an outbound call
tcccSDK->call(callParams, startCallCallbackImpl);
// End the call
tcccSDK->terminate();
```

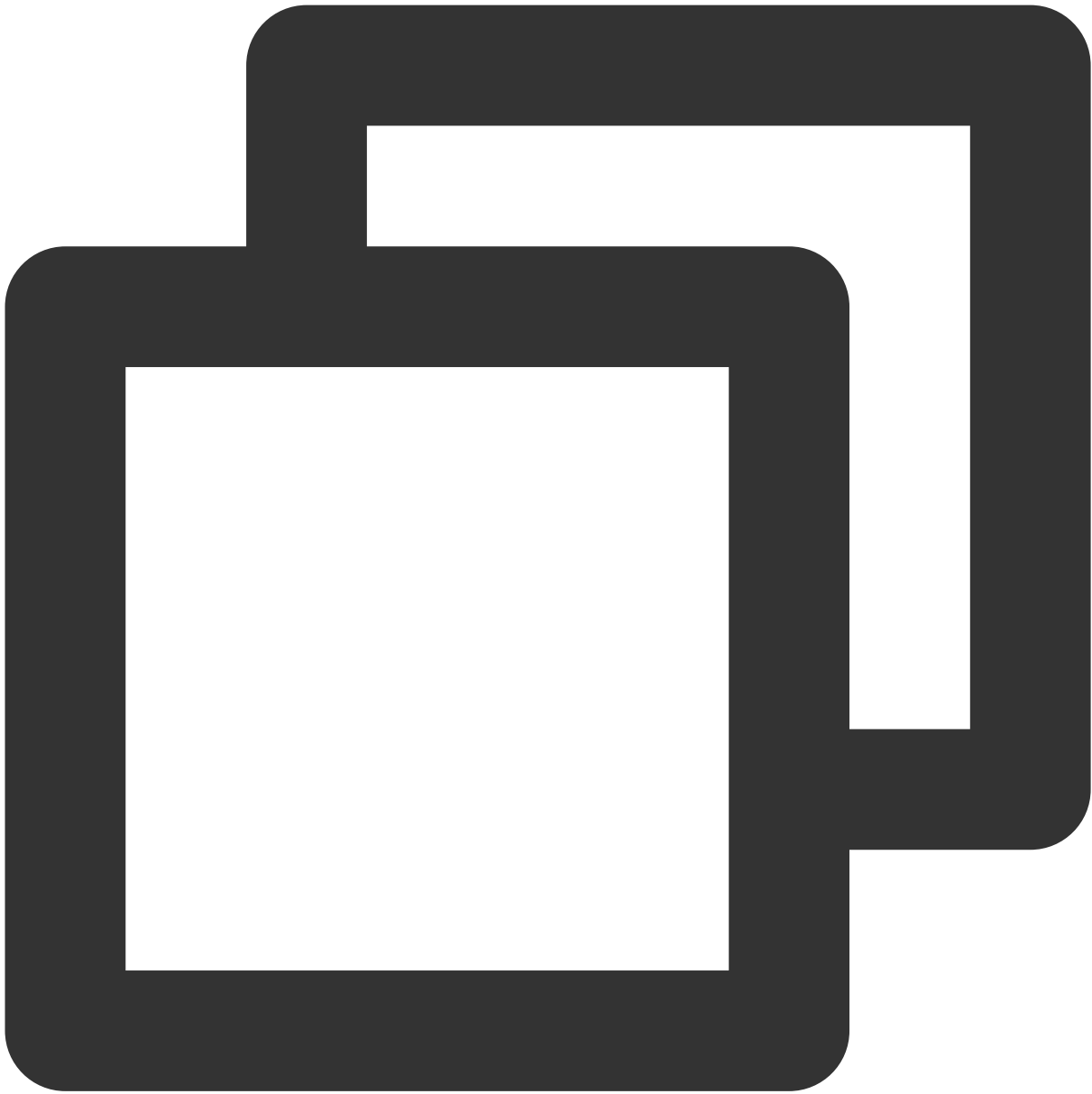
Audio Device API Functions

API	Description
setAudioCaptureVolume	Sets the local audio capture volume.
getAudioCaptureVolume	Obtains the local audio capture volume.
setAudioPlayOutVolume	Sets the remote audio playback volume.
getAudioPlayOutVolume	Obtains the remote audio playback volume.
setAudioRoute	Sets audio routing.

Debugging APIs

API	Description
getSDKVersion	Obtains the SDK version.
setLogLevel	Sets the log output level.
setConsoleEnabled	Enables/Disables console log print.
callExperimentalAPI	Calls an experimental API.

Sample Code for Obtaining SDK Version



```
// Obtain SDK version number
tcccSDK->getSDKVersion();
```

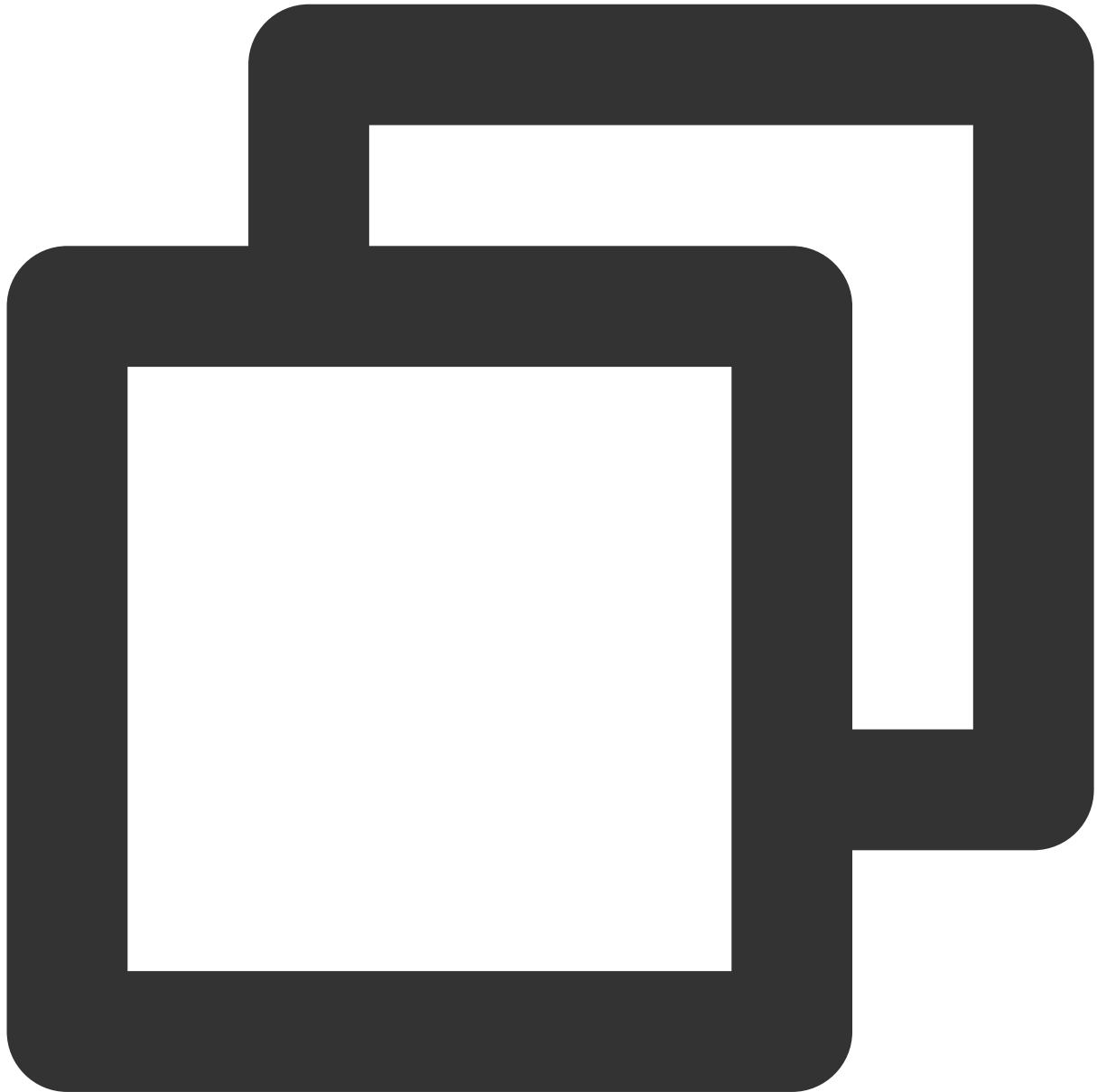
Error and Warning Events

API	Description
onError	Error event callback

[onWarning](#)

Warning event callback

Sample Code for Handling Error Event Callbacks



```
// Set the callback. TCCallbackImpl needs to be derived from ITCCallback
class TCCallbackImpl:public ITCCallback {
public:
    TCCallbackImpl() {}
    ~TCCallbackImpl() {}
    // Error event callback
```

```
void onError(TCCCErrCode errCode, const char* errMsg, void* extraInfo) {}  
// Warning event callback  
void onWarning(TCCCCWarning warningCode, const char* warningMsg, void* extraInfo)  
{  
};  
TCCCCallbackImpl* tcccCallback = new TCCCCallbackImpl();  
tcccSDK->addCallback(tcccCallback);
```

Call Event Callback

API	Description
onNewSession	New session event, including inbound and outbound calls
onEnded	Session end event
onAudioVolume	Callback for volume feedback
onNetworkQuality	Real-time statistics callback of network quality

Sample Code for Answer and Agent Hang-up Event Callbacks



```
// Set the callback. TCCallbackImpl needs to be derived from ITCCallback
class TCCallbackImpl:public ITCCallback {
public:
    TCCallbackImpl() {}
    ~TCCallbackImpl() {}
    // New session event, including inbound and outbound calls
    void onNewSession(TCCSessionInfo info) {}
    // Session end event
    void onEnded(EndedReason reason, const char* reasonMessage, const char* sessionid) {}
};
```

```
TCCCCallbackImpl* tcccCallback = new TCCCCallbackImpl();  
tcccSDK->addCallback(tcccCallback);
```

Event Callback of Connection with Cloud

API	Description
onConnectionLost	The connection between the SDK and the cloud has been disconnected.
onTryToReconnect	The SDK is trying to reconnect to the cloud.
onConnectionRecovery	The connection between the SDK and the cloud has been restored.

Sample Code for Event Callback of Connection with Cloud



```
// Set the callback. TCCallbackImpl needs to be derived from ITCCallback
class TCCallbackImpl:public ITCCallback {
public:
    TCCallbackImpl() {}
    ~TCCallbackImpl() {}
    // The connection between the SDK and the cloud has been disconnected
    void onConnectionLost(TCCServerType serverType) {}
    // The SDK is trying to reconnect to the cloud
    void onTryToReconnect(TCCServerType serverType) {}
    // The SDK's connection with the cloud has been restored
    void onConnectionRecovery(TCCServerType serverType) {}
```

```
};  
TCCCCallbackImpl* tcccCallback = new TCCCCallbackImpl();  
tcccSDK->addCallback(tcccCallback);
```

API Error Codes

Basic Error Codes

Symbol	Value	Meaning
ERR_SIP_SUCCESS	200	Execution succeeded.
ERR_UNRIGIST_FAILURE	20001	Login failed.
ERR_ANSWER_FAILURE	20002	Failed to answer the call, usually because the TRTC failed to enter the room.
ERR_SIPURI_WRONGFORMAT	20003	URI format error.
ERR_HTTP_REQUEST_FAILURE	-10001	HTTP request failed. Please check your network connection.
ERR_HTTP_TOKEN_ERROR	-10002	The token login ticket is incorrect or has expired.
ERR_HTTP_GETSIPINFO_ERROR	-10003	Failed to obtain the agent configuration. Please contact us.

SIP Error Codes

Symbol	Value	Meaning
ERR_SIP_BAD_REQUEST	400	Error request.
ERR_SIP_UNAUTHORIZED	401	Unauthorized (username or password is incorrect).
ERR_SIP_AUTHENTICATION_REQUIRED	407	Proxy authentication required. Please check whether the login API has been called.
ERR_SIP_REQUESTTIMEOUT	408	Request timeout (network timeout).
ERR_SIP_REQUEST_TERMINATED	487	Request termination (network error, in case of network interruption).
ERR_SIP_SERVICE_UNAVAILABLE	503	Service not available.

ERR_SIP_SERVER_TIMEOUT	504	Service timeout.
------------------------	-----	------------------

Audio Device Error Codes

Symbol	Value	Meaning
ERR_MIC_START_FAIL	-1302	Failed to start the microphone. The device's microphone configuration program (driver) is abnormal. Please disable and re-enable the device, restart the device, or update the configuration program.
ERR_MIC_NOT_AUTHORIZED	-1317	No access to the microphone. This usually occurs on mobile devices and may be because the user denied the access.
ERR_MIC_SET_PARAM_FAIL	-1318	Failed to set microphone parameters.
ERR_MIC_OCCUPY	-1319	The microphone is occupied. This occurs when, for example, the user is currently having a call on the mobile device.
ERR_MIC_STOP_FAIL	-1320	Failed to stop the microphone.
ERR_SPEAKER_START_FAIL	-1321	Failed to start the speaker, for example, on Windows or Mac.
ERR_SPEAKER_SET_PARAM_FAIL	-1322	Failed to set speaker parameters.
ERR_SPEAKER_STOP_FAIL	-1323	Failed to stop the speaker.
ERR_UNSUPPORTED_SAMPLERATE	-1306	Unsupported audio sample rate.

Network Error Codes

Symbol	Value	Meaning
ERR_RTC_ENTER_ROOM_FAILED	-3301	Failed to enter the room. Please view -3301 in onError to confirm the message for the reason of the failure.
ERR_RTC_REQUEST_IP_TIMEOUT	-3307	Request for IP and Sig timed out. Please check whether your network is functioning properly and whether UDP is unblocked in your network firewall.
ERR_RTC_CONNECT_SERVER_TIMEOUT	-3308	Request for room entry timed out. Please check your network connection or whether you are on a

		VPN. You can also try switching to 4G for confirmation.
ERR_RTC_ENTER_ROOM_REFUSED	-3340	Room entry request was denied. Please check whether you are continually calling enterRoom to enter the room of the same ID.

常见问题

FAQs About Web SDK

Last updated : 2024-04-01 17:55:54

What frameworks does the Web SDK support?

The Web SDK is implemented in pure JavaScript and supports running in environments such as Vue, React, uni-app, PHP, and JSP.

Can the SDK interface display other information?

It is not supported.

Can the call bar button in the SDK be hidden?

Supported.

What is UserId when the SDK is initialized?

UserId refers to the account in Cloud Contact Center, usually in the format of an email address. It can be created from the console or the management background.

How do I switch accounts with the SDK?

By reinitializing the SDK with a different UserId, the accounts can be automatically switched.

Why does the SDK need to use HTTPS in the deployment page?

Due to browser restrictions, microphone access can only be obtained under HTTPS.

How do I specify the display number when making an outbound call?

It is not supported on the interface. You can specify the display number when calling the [outbound API](#) of the SDK.

Will the token need to be renewed? What if it expires?

After the SDK initialization is complete, the Token will not need to be renewed. Please make sure the Token is valid while initializing the SDK.

Why is a device error is prompted after login?

1. Check whether the website URL is HTTPS.
2. Check whether the microphone access is allowed.
3. Refer to [Detecting Websites](#), and follow the steps.

4. Developers can make a custom prompt according to the API provided by the SDK, `isBrowserSupported` and `isEnvSupported`.

Hint:

Microphone: Microphone device error. Please check your device.
The microphone is unavailable. Incoming calls and audio/video calls will not capture the user's voice

Device Check

Cancel

Why is there no ring when a call comes in?

If during an incoming call, the SDK's page is minimized or switched to another page, there may be no ring due to [browser's restrictions](#). It is recommended to turn on browser notifications or listen to the SDK's `callIn` event, and have the business side make a strong prompt.

Why does the outbound call fail?

After the SDK initialization is complete, wait for the ready event before making an outbound call. In addition, ensure that there are numbers that can make outbound calls in the list of instances.

Why is the call interrupted suddenly?

Check the `closeBy` field of the SDK's `sessionEnded` event to determine who hung up.

FAQs About uni-app SDK

Last updated : 2024-04-01 17:57:11

How do I view Cloud Contact Center logs?

The logs of Cloud Contact Center are compressed and encrypted by default, with suffix .log.

Log path:

Android: `/sdcard/Android/data/package name/files/tccc`

iOS: In the **tccc** folder under the **sandbox/Documents** directory

Does the Cloud Contact Center SDK support X86 simulators on Android?

The current version of Cloud Contact Center does not support simulators, but it will support simulators in the future. If you need to run on a simulator, we recommend debugging on an iOS x86 simulator.

Does the Cloud Contact Center SDK support the CPU type armv7 on iOS?

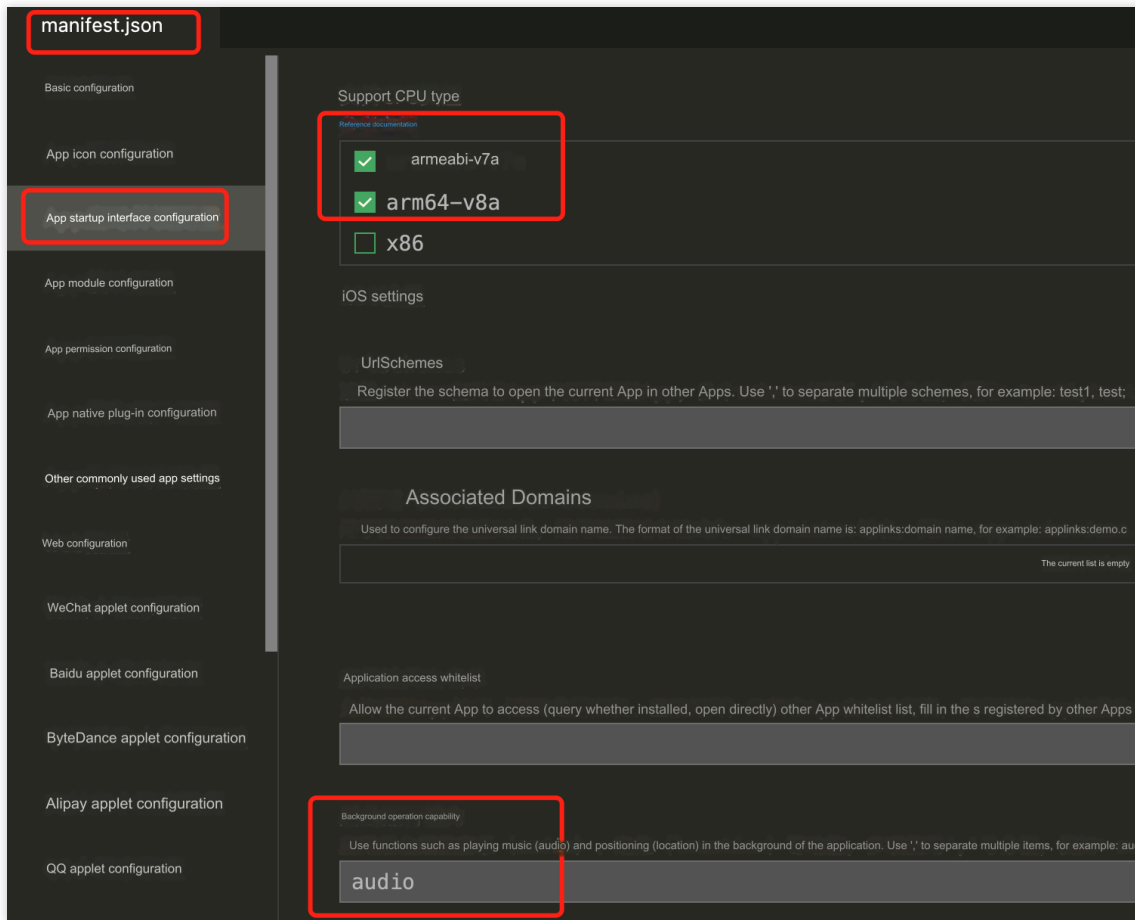
Because only iPhone 5c and earlier versions have this type of CPU, and almost no one is using it now. Therefore, we do not adapt to this type of CPU. When packaging iOS on the cloud, you need to modify the **manifest.json** file.



```
"validArchitectures": [  
    "arm64"  
],
```

Why do phone calls get interrupted when iOS switches to the background?

When the mobile application switches to the background, the operating system will suspend the application process to save resources. You can configure **audio background mode** in iOS to ensure that the application will not be terminated when there is audio impact.



Why can't incoming calls be handled under mobile phones?

If a new session occurs while the mobile phone is running in the foreground, you will receive **onNewSession** callback. However, we do not recommend handling incoming calls on mobile phones (the app will pause when switching to the background). We recommend you activate the feature of receiving calls on mobile phones.

FAQs About Client SDK

Last updated : 2024-04-01 17:58:36

How do I view Cloud Contact Center logs?

The logs of Cloud Contact Center are compressed and encrypted by default, with suffix .log.

Android log path: `/sdcard/Android/data/package name/files/tccc`

iOS log path: `sandbox/Documents/tccc`

Does TCCC Agent Android support emulators?

The current version of TCCC does not support simulators, but it will support them in the future.

Why does audio collection stop when Android is in the background?

Android 9.0 limits microphone access when an app goes into the background to prevent calls from being muted. To avoid this, send a foreground notification when the app is in the background, or use Settings to keep the screen on.

Are the callbacks on iOS all on the main thread?

All callbacks in the Swift and OC interfaces are on the main thread, so developers do not need to handle them specially. However, callbacks in c++ are not on the main thread and need to be assessed by the business layer and then switched to the main thread:



```
if ([NSThread isMainThread]) {  
    // On the main thread, you can directly process  
    return;  
}  
dispatch_async(dispatch_get_main_queue(), ^{  
    // Callbacks are made from a non-main thread.  
});
```

Is there a corresponding SDK for other platforms like Windows?

TCCC provides a cross-platform SDK. If needed, you can [Contact Us](#), and we will provide it offline.

Integrated Telephony Customer Service Implementing One-Click Outbound Call Web

Last updated : 2024-04-01 18:09:01

Step 1: Initialize the SDK

Please refer to [Initializing SDK](#).

Note

The following steps should be performed after the 'tccc.events.ready' event is successful.

Step 2: Implement Clicking Button to Trigger SDK Outbound Call

Vue

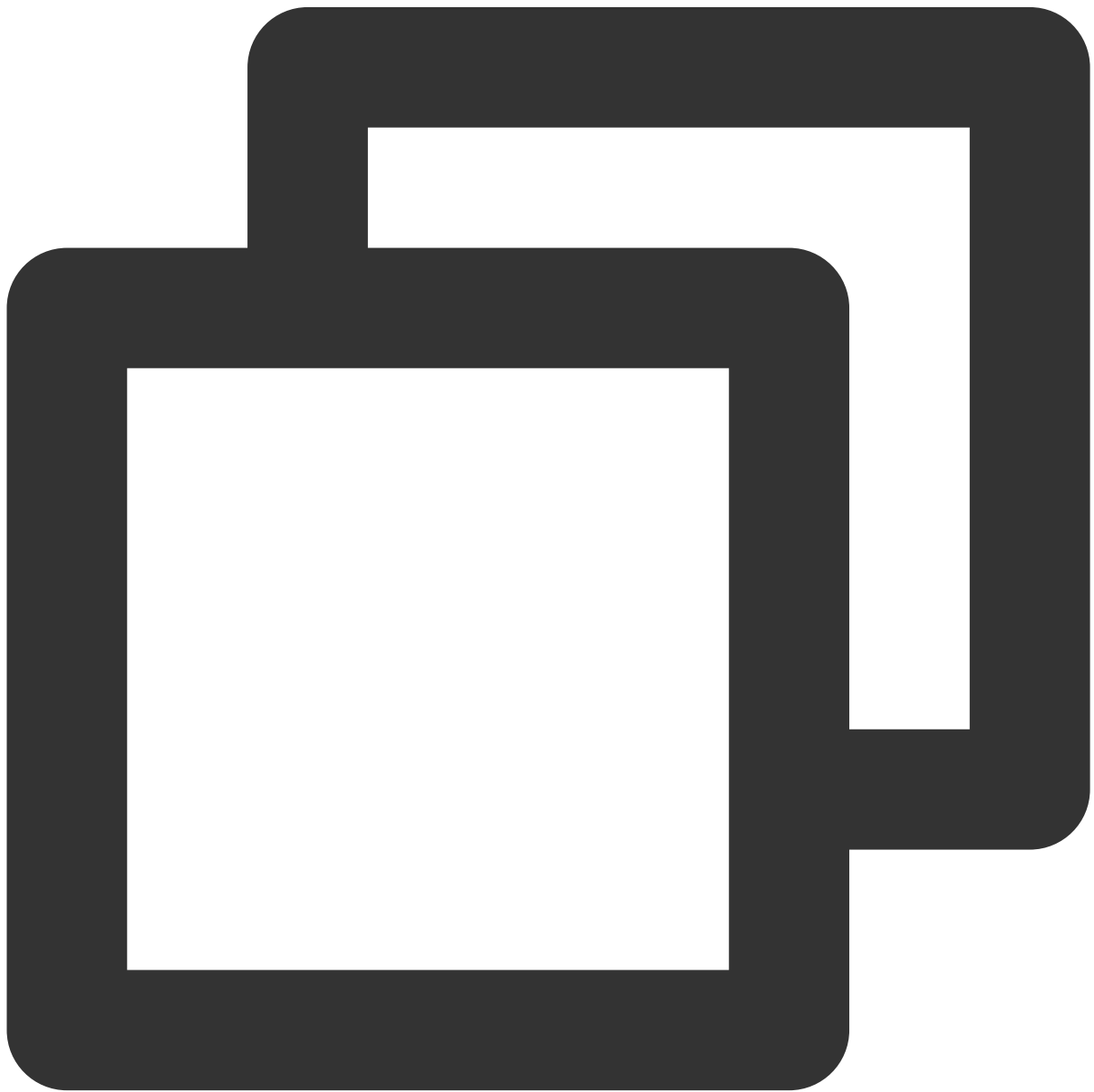
React

Native JS



```
<template>
  <button @click="sdkCall">One-click outbound call</button>
</template>
<script>
export default {
  data() {
    phoneNumber: '19999999999' // Replace it with a real outbound number
  },
  methods: {
    sdkCall() {
      window.tccc.Call.startOutboundCall({
```

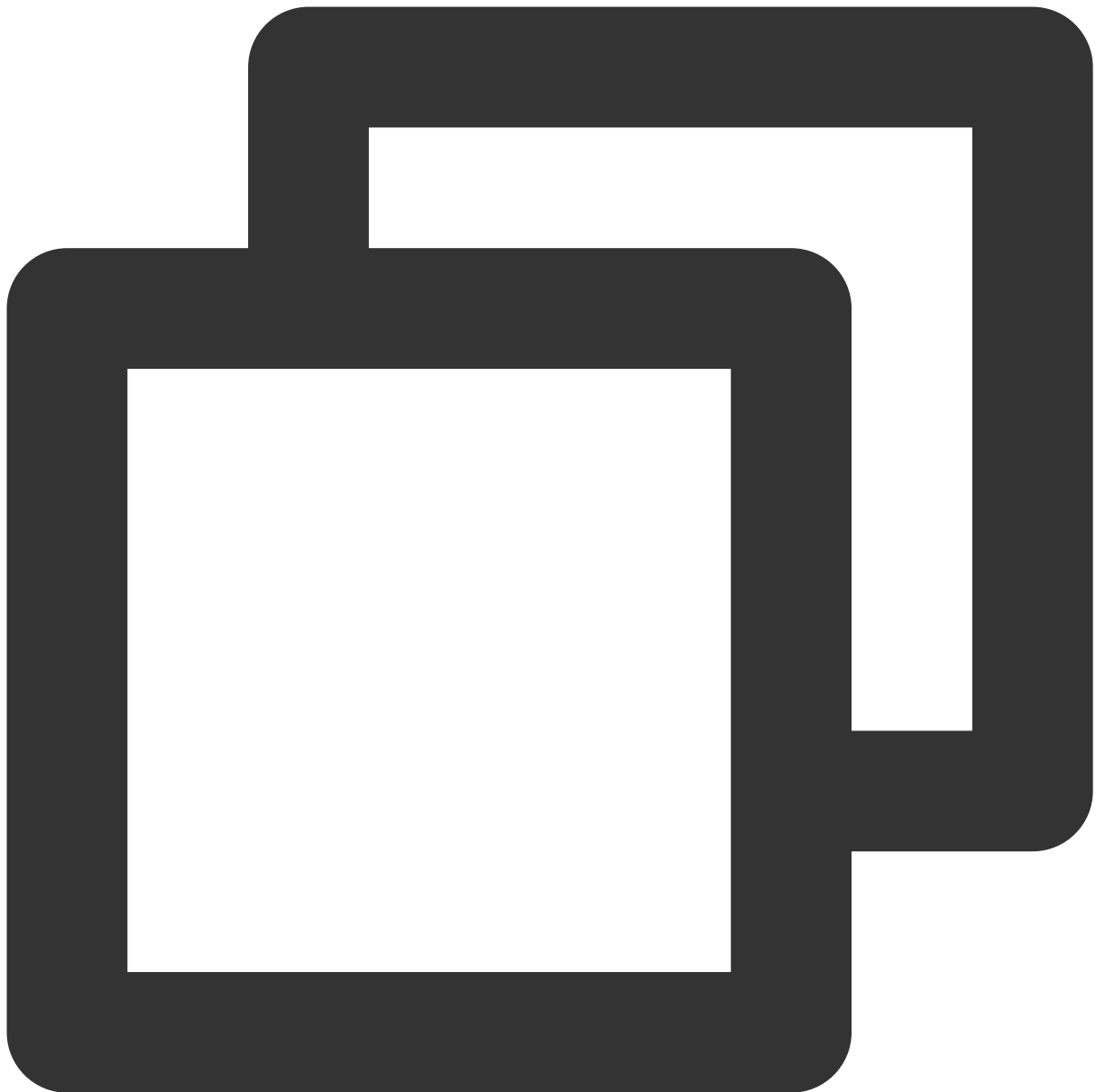
```
        phoneNumber: this.phoneNumber,  
    }).then((res) => {  
        this.sessionId = res.data.sessionId;  
    }).catch((err) => {  
        const error = err.errorMsg;  
    })  
    }  
}  
}  
</script>
```



```
import { useState } from 'react';
export function CallButton() {
  const [phoneNumber, setPhoneNumber] = useState('19999999999') // Replace it with

  function sdkCall(phoneNumber) {
    window.tccc.Call.startOutboundCall({
      phoneNumber,
    }).then((res) => {
      this.sessionId = res.data.sessionId;
    }).catch((err) => {
      const error = err.errorMsg;
    })
  }

  return (
    <button onClick={sdkCall}>One-click outbound call</button>
  )
}
```

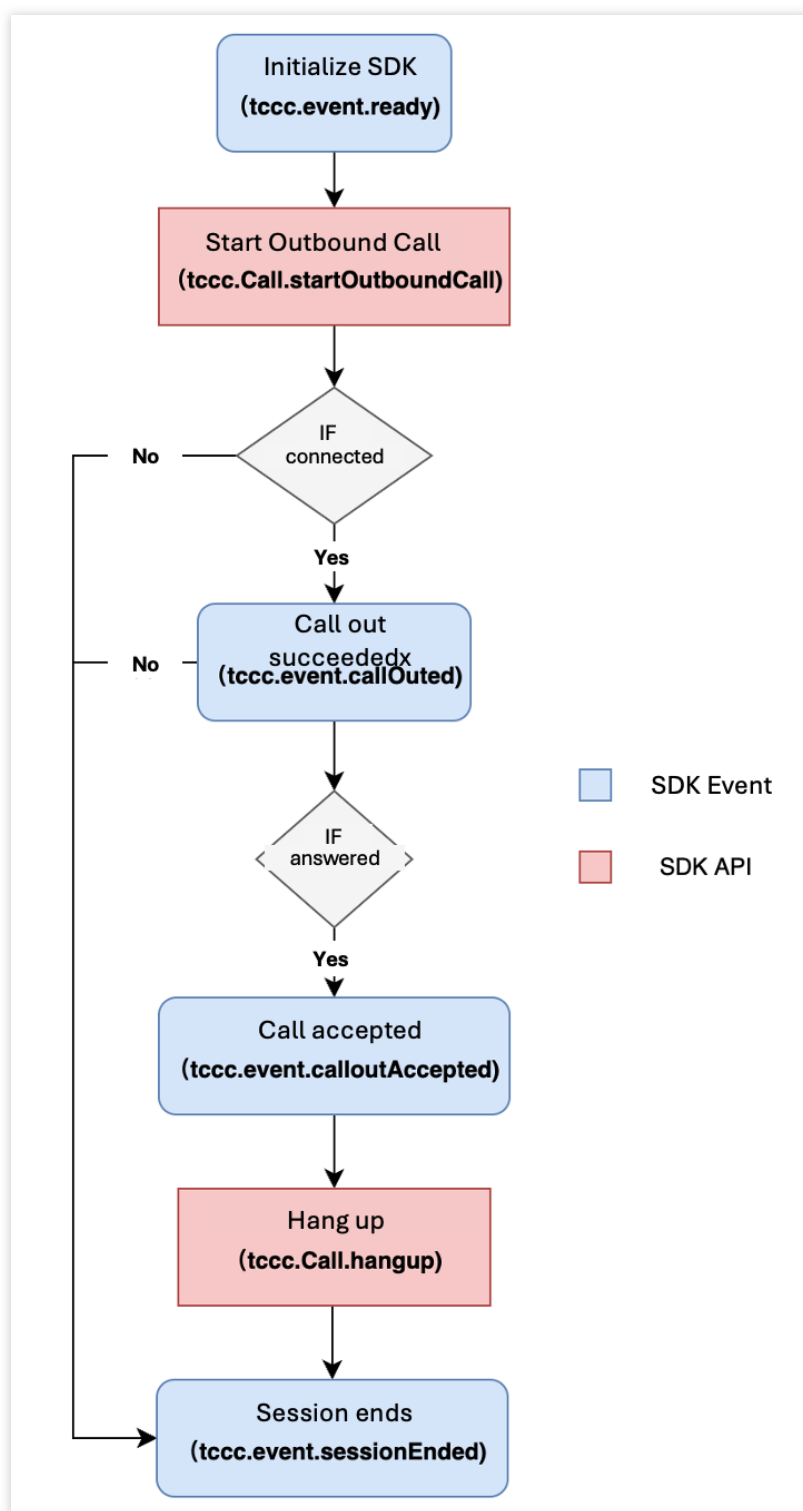


```
<button id="call">One-click outbound call</button>
<script>
  function sdkCall(phoneNumber) {
    window.tccc.Call.startOutboundCall({
      phoneNumber,           // Outbound number
      phoneDesc: 'Tencent'   // Remarks, which will replace the display of th
    }).then((res) => {
      // Outbound call succeeded. Obtain the outbound ID, which can be used t
      const sessionId = res.data.sessionId
    }).catch((err) => {
      // Outbound call failed. Obtain the failure reason for prompt
```

```
        console.error(err.errMsg)
    })
}
// Listen to the click event of the button and trigger the outbound call method
document.getElementById('call').addEventListener('click', () => {
    // Replace it with a real outbound number
    sdkCall('19999999999');
})
</script>
```

After the outbound call is successfully triggered, wait for the other party to answer and trigger related events in turn.

Outbound Call Event Process



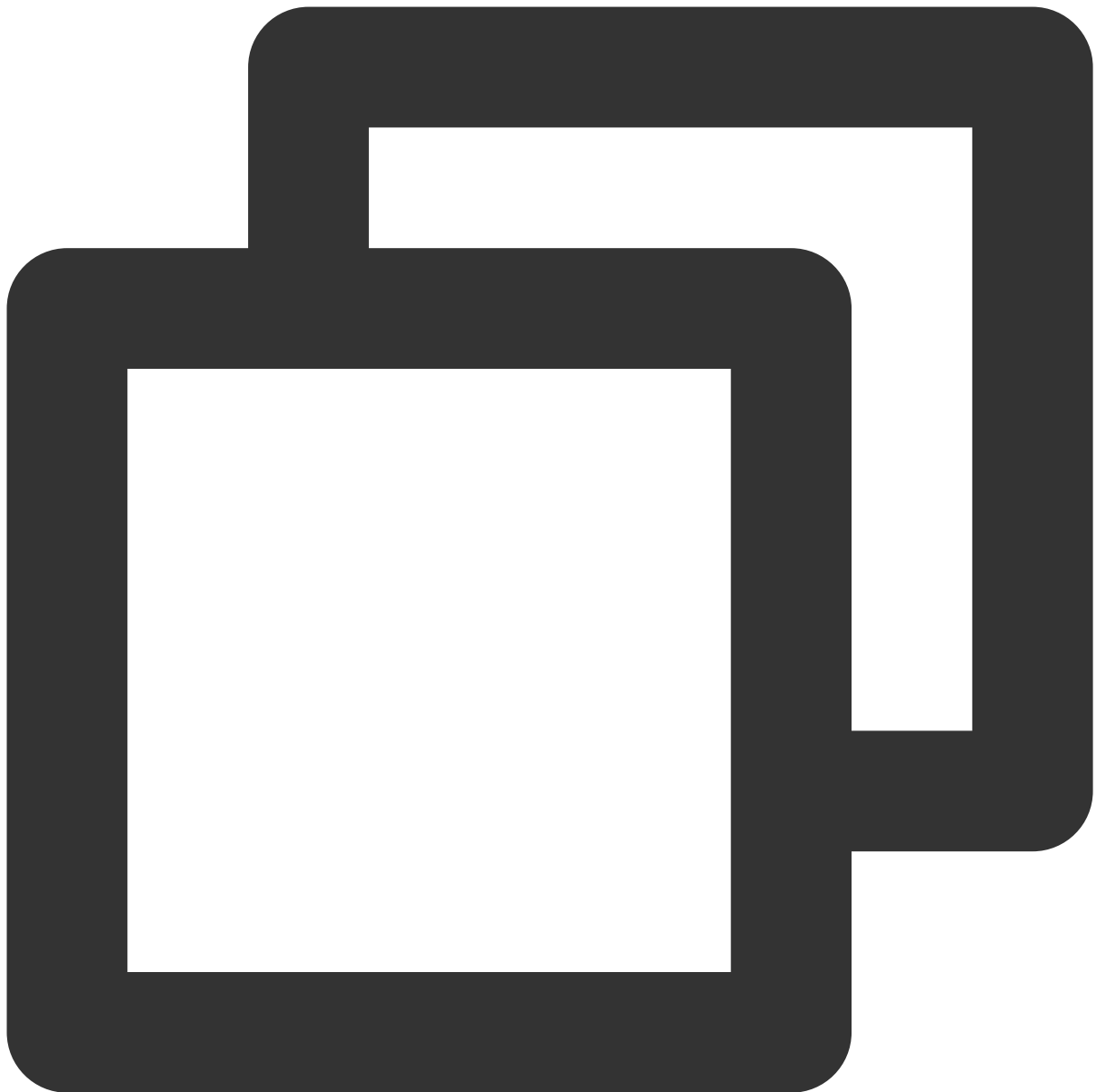
uni-app

Last updated : 2024-04-01 18:09:28

Call-related API Functions

API	Description
call	Initiates a call
answer	Answers the inbound call
terminate	Ends the call
sendDTMF	Sends Dual-Tone Multi-Frequency (DTMF) signals
mute	Mutes.
unmute	Unmutes.
startPlayMusic	Starts playing music
stopPlayMusic	Stops playing music

Sample Code for Initiating and Ending a Call



```
// Initiate a call. Call the login API before initiating a call. tcccSDK.login
tcccSDK.call({
  to: '134xxxx',           // Contact number (required)
  remark: "xxx",           // Number remarks, which will replace the number displayed
  uui: "xxxx",             // User-defined data (optional)
}, (code,message) => {
  if (code == TcccErrorCode.ERR_NONE) {
    // Initiation succeeded
  } else {
    // Initiation failed
  }
})
```

```
});  
// End the call  
tcccSDK.terminate();  
// Answer the call  
tcccSDK.answer((code,message) => {  
    if (code == TcccErrorCode.ERR_NONE) {  
        // Answer succeeded  
    } else {  
        // Answer failed  
    }  
});
```

Android

Last updated : 2024-04-01 18:10:07

Call-related API Functions

API	Description
call	Initiates a call
answer	Answers the inbound call
terminate	Ends the call
sendDTMF	Sends Dual-Tone Multi-Frequency (DTMF) signals
mute	Mutes.
unmute	Unmutes.

Sample Code for Initiating and Ending a Call



```
TCCTypeDef.TCCCStartCallParams callParams =new TCCTypeDef.TCCCStartCallParams();  
//1343xxxx is the phone number  
callParams.to = "13430xxxx";  
// Initiate a call. Call the login API before initiating a call. tcccSDK.login  
tcccSDK.call(callParams, new TXCallback() {  
    @Override  
    public void onSuccess() {  
        // call success  
    }  
  
    @Override
```

```
public void onError(int code, String desc) {  
    // call error  
}  
});  
// End the call  
tcccSDK.terminate("");
```

IOS

Last updated : 2024-04-01 18:10:27

Call-related API Functions

API	Description
call	Initiates a call
answer	Answers the inbound call
terminate	Ends the call
sendDTMF	Sends Dual-Tone Multi-Frequency (DTMF) signals
mute	Mutes.
unmute	Unmutes.

Sample Code for Initiating and Ending a Call



```
class TCCCCCommonCallback : public ITXCallback {
private:
    NSString* mFunName;
public:
    TCCCCCommonCallback(NSString* funName) {
        mFunName = funName;
    }
    ~TCCCCCommonCallback() override {

    }
    void OnSuccess() override {
```



```
        // Succeeded
    }

    void OnError(TCCCErrors error_code, const char *error_message) override {
        std::string copyErrMsg = makeString(error_message);
        // Failed
    }
};

TCCCommonCallback* startCallCallbackImpl = nullptr;
if (nullptr == startCallCallbackImpl) {
    startCallCallbackImpl = new TCCCommonCallback(@"startCall");
}

TCCStartCallParams callParams;
// Phone number for the call
callParams.to = "";
// Initiate a call. Call the login API before initiating a call. tcccSDK->login
tcccSDK->call(callParams, startCallCallbackImpl);
// End the call
tcccSDK->terminate();
```

Implementing Call-In Function Web

Last updated : 2024-04-01 18:12:41

Initializing SDK

Please refer to [Initializing SDK](#).

Note:

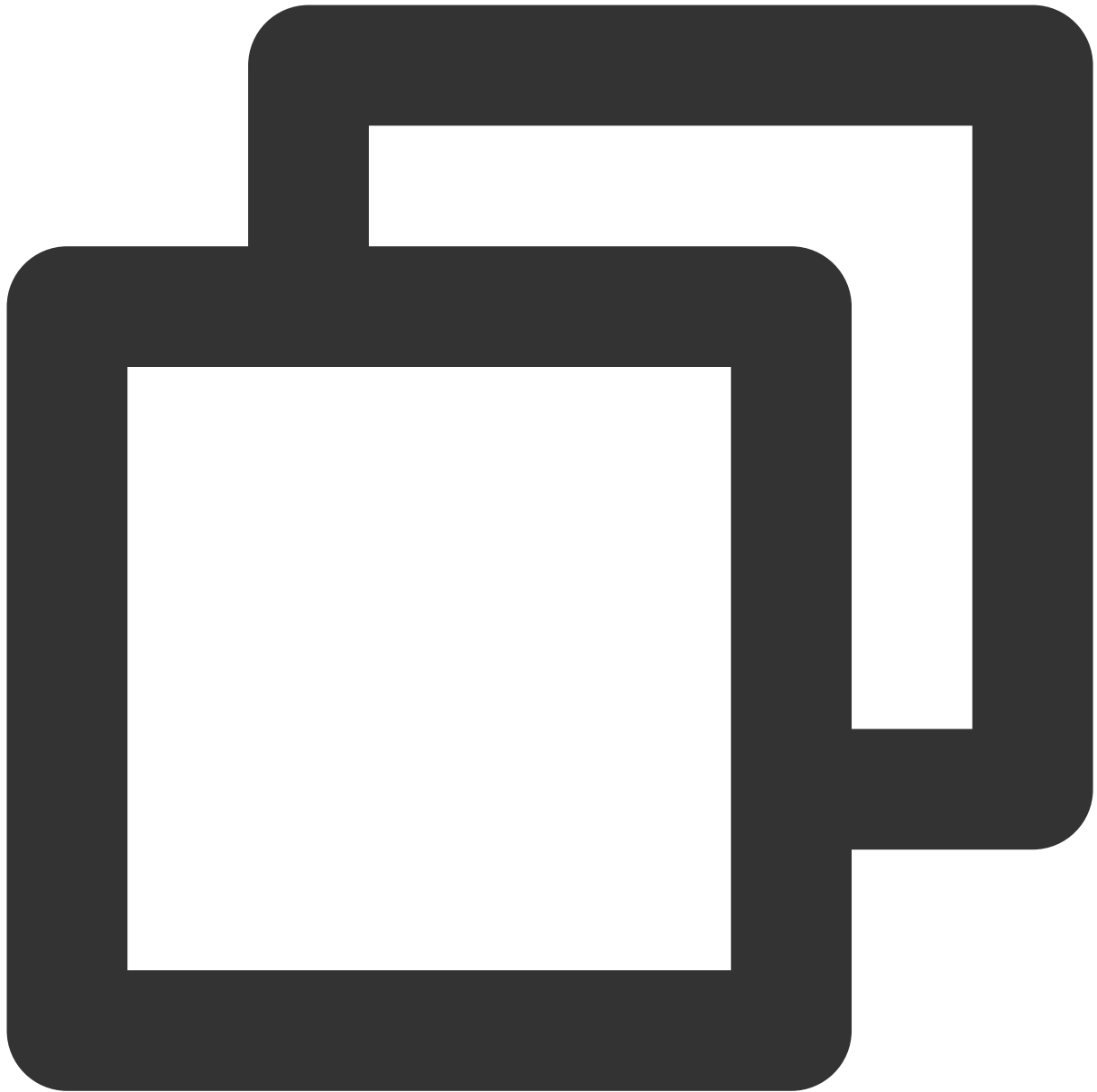
The following steps should be performed after the 'tccc.events.ready' event is successful.

Answering Methods

Method 1: Answer via SDK API

1. Bind the inbound call event `tccc.events.callIn` through `tccc.on` to monitor the inbound call and obtain sessionId;
2. Use `tccc.Call.accept()` to answer actively.

Sample code:



```
let sessionId; //Exists in the public zone, and can be conveniently used at any time

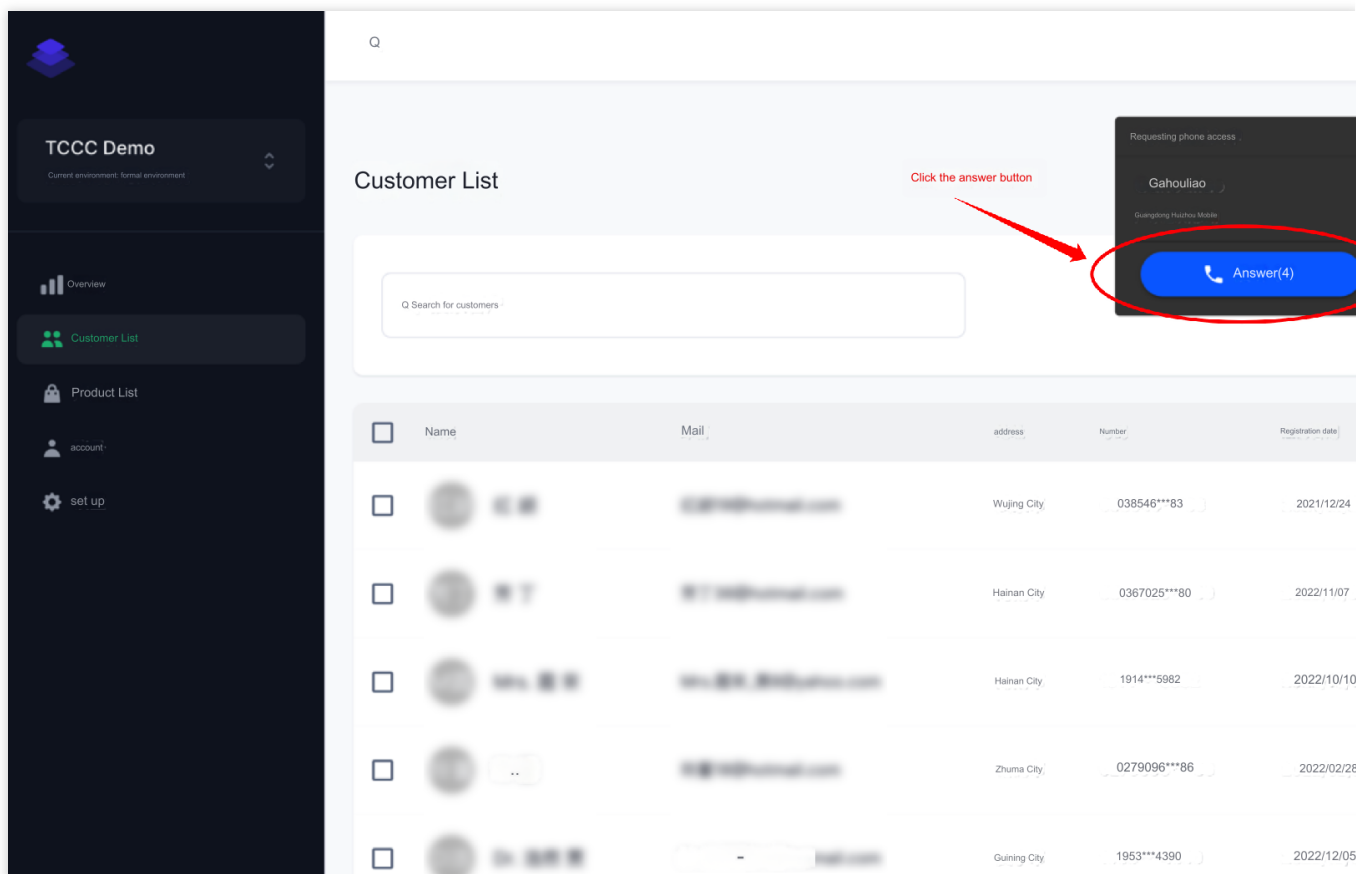
// Monitors inbound call events
window.tccc.on(window.tccc.events.callIn, (response) => {
  // Triggered when a session calls in, and stores this session's sessionId in a public variable
  sessionId = response.data.sessionId;
})

// Implements the answering method
function accept() {
  if (sessionId) {
```

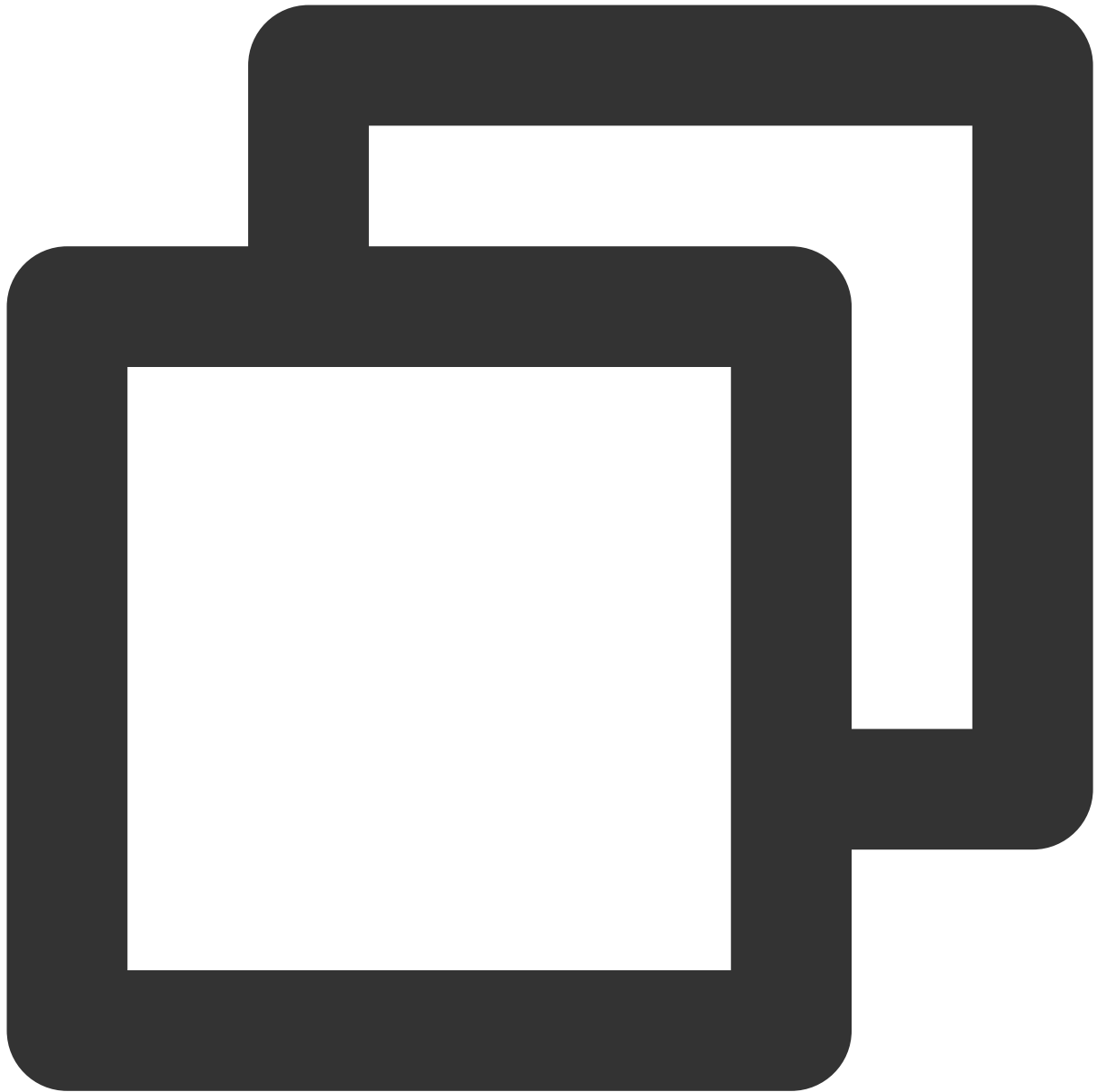
```
window.tccc.Call.accept({ sessionId })
  .then(() => {
    // Successfully answers and starts the call
  })
  .catch(err => {
    // Failed to answer, and displays detailed error reason
    const error = err.errorMsg;
  })
} else {
  console.error('The session to be answered was not found');
}
}
```

// Then, accept() can be executed at the required place to trigger answering the call

Method 2: Answer by Clicking on the Call Bar



Other Related Events



```
window.tccc.on(window.tccc.events.callIn, (response) => {  
  // Triggered when session calls in  
})  
window.tccc.on(window.tccc.events.userAccessed, (response) => {  
  // Agent connection  
})  
window.tccc.on(window.tccc.events.sessionEnded, (response) => {  
  // Triggered when session ends  
})
```