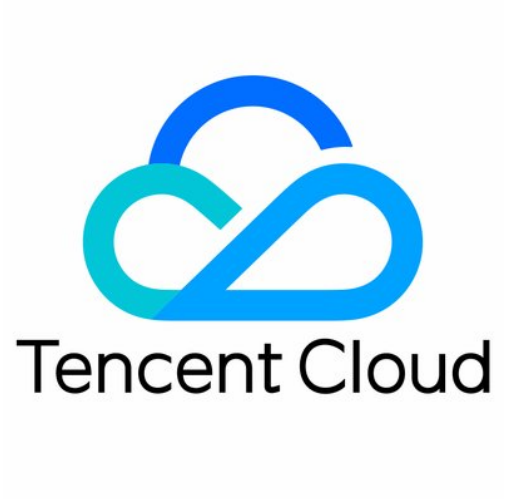
Cloud Contact Center

Developer Guide

Product Documentation



Contents

Developer Guide
SDK Development Guide
Getting Started

Last updated：2025-01-09 15:23:21

Tencent Cloud Contact Center helps you set up a customer contact center with voice, chat, audio and video capabilities. This guide explains how to use the provided SDKs to implement into your system.

Get Started

You can follow these steps for integrated development:

Step	Operation
1	Creating Cloud Contact Center application
2	Inbound / Outbound Configuration Guide Outbound Call Quick Set Up Inbound Call Quick Set Up
3	Refer to Integrating Agent-side SDK to integrate the agent side into your own system
4	Omni SDKs support different terminals including IOS, Android, Uniapp, Web SDK, please refer to the link for more (It's more convenient to navigate from the sidebar) Omni SDK for Inbound and Outbound

Contact Us

Join our WeCom Group Community:

Come and Join our Whatsapp Community

Come and Join our Discord Community

Integrating Agent SDK
Demo Quick Run
Web

Last updated：2025-01-09 15:23:21

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.

Contact Us

Join our WeCom Group:

Come and Join our Whatsapp Community

Come and Join our Discord Community

Android

Last updated：2025-01-09 15:23:21

Android Demo Quick Run

Tencent Cloud Contact Center Android SDK enables agents to handle calls via PC, SIP phones, or mobile.
This guide shows how to quickly set up and run the Android demo.

Developer Environment Requirements

Android Studio 3.5+。
Android 4.1 (SDK API 16) or later.

Prerequisite

Login Tencent Cloud Console
You have activated Cloud Contact Center application.
Completed Bring Your Own Carrier via SIP Trunk and finished IVR management.

Key Concepts

SdkAppId: A unique application ID created in Cloud Contact Center console. Each Tencent Cloud account can create up to 20 TCCC applications.
UserID: The member ID added in the Tencent Cloud Contact Center, usually an email address. For first-time setup, the admin account and password can be found in the Internal Message (sub-accounts must subscribe to TCCC product notification). Each SDKAppID supports multiple UserIDs, and additional licenses can be purchased atAgent Purchase if needed.
SecretId and SecretKey: Developers need credentials to call the Cloud API. SecretId and Secretkey can be created in the Tencent Cloud console.
Token: A login credential obtained viaCreateSDKLoginToken API. The app requests real-time tokens from your server, which stores the generation code and encryption key.

Steps

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

Souce Code: tccc-agent-java-example

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

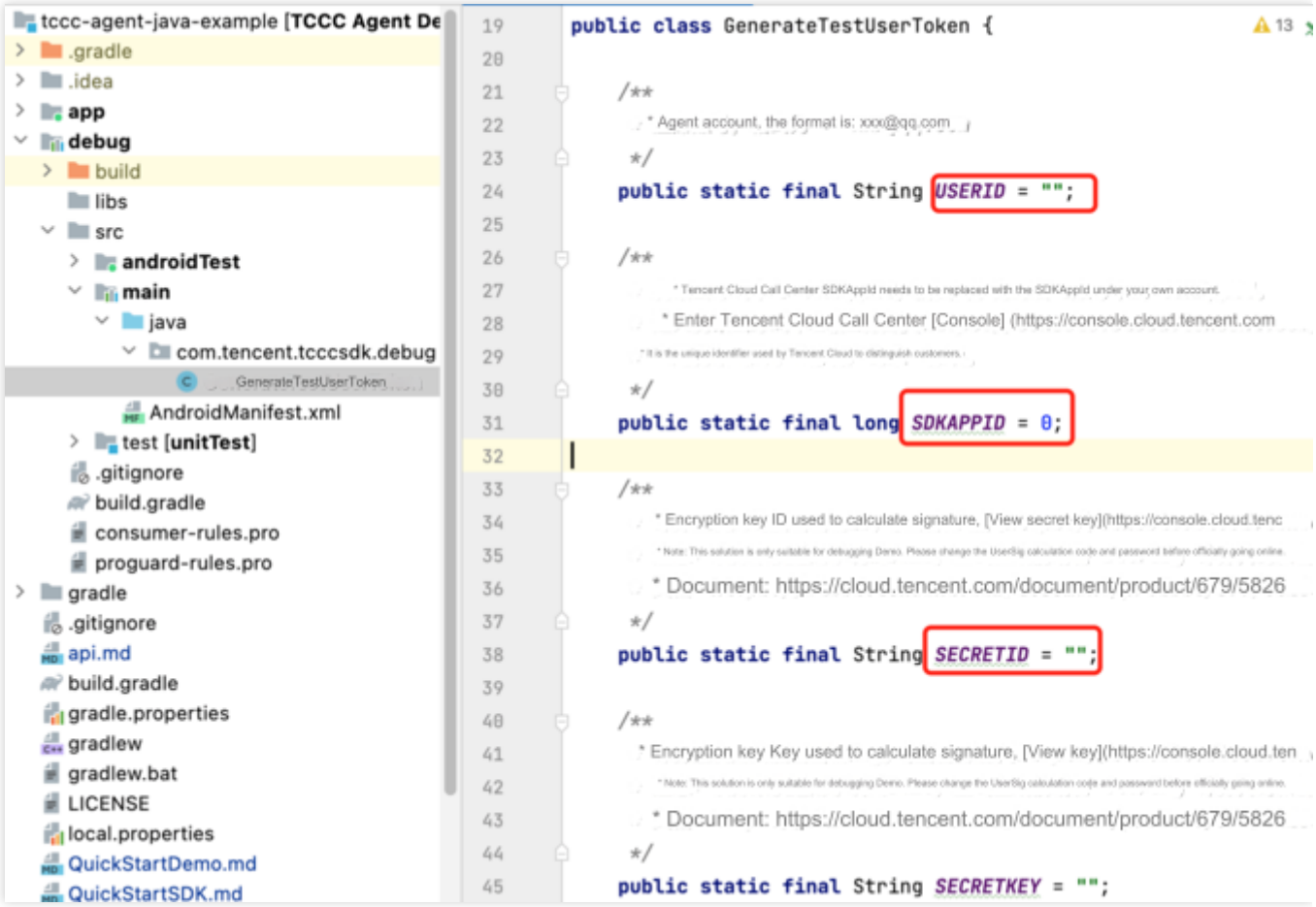
1. Open `debug/src/main/java/com/tencent/tcccsdk/debug/GenerateTestUserToken.java`
2. Set parameters in GenerateTestUserToken.java file:

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

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

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

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



Caution:

Do not include the token generation code in your app's production version for the following reasons:

The provided code is for testing basic SDK features only and is not secure for production. SECRETKEY in client-side code can be easily reverse-engineered, and web code is even more vulnerable. If your key is leaked, attackers can generate valid tokens and misuse your Tencent Cloud resources.

Safer approach is to place the token generation code and encryption key on your server. The app should request real-time tokens from your server as needed. This method is more secure, as breaching a server is significantly harder than cracking a client-side app. For more details, see Creating SDK Login Token.

Step 3: Compile and Run the Demo

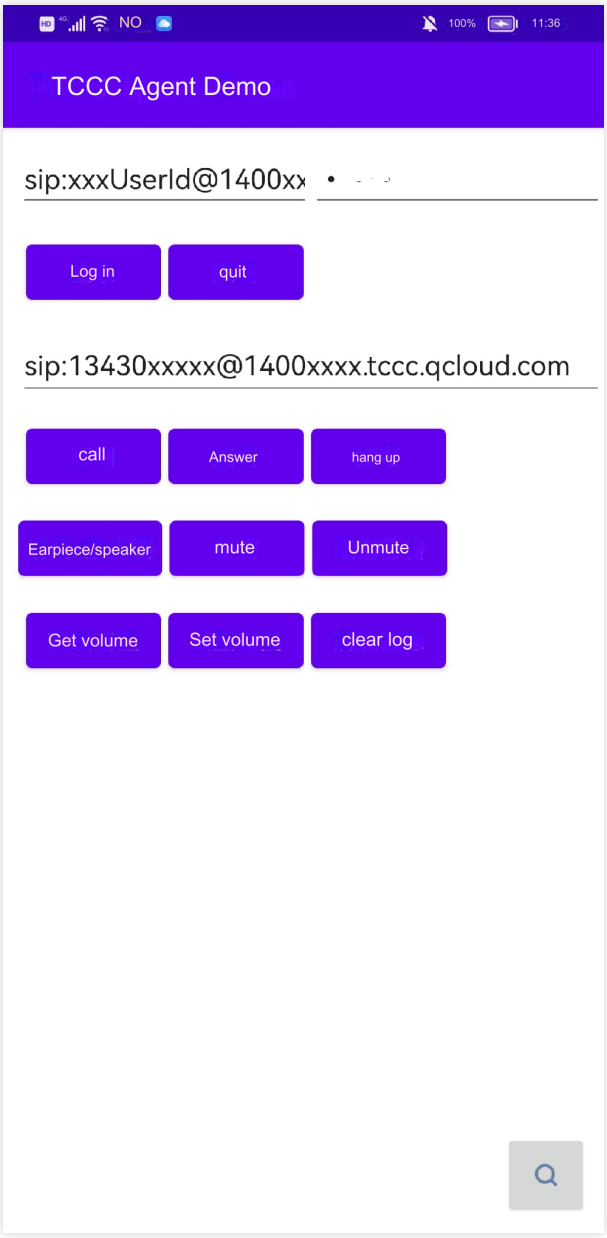
Open source code project `tccc-agent-java-example` in Android Studio (version 3.5 or later), click **Run**.

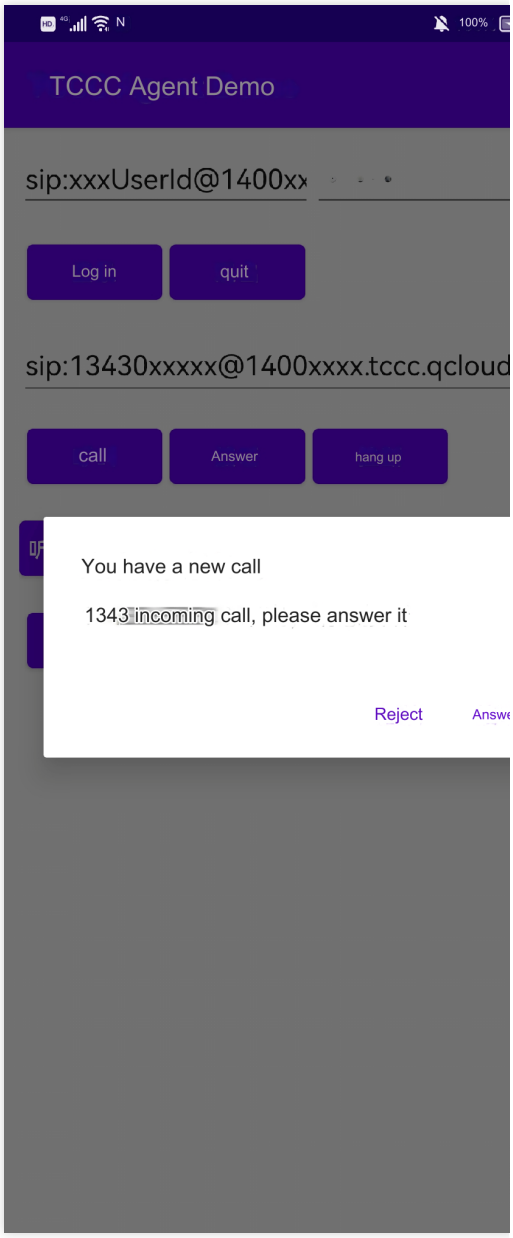
1. **Log In.**
2. Enter the phone number and start a call.

Showcase

Basic features as shown below:

Call Example	Call Answer Example


NO
100%
11:36
TCCC Agent Demo
sip:xxxUserId@1400xx
Log in
quit
sip:13430xxxxx@1400xxxx.tccc.qcloud.com
call
Answer
hang up
Earpiece/speaker
mute
Unmute
Get volume
Set volume
clear log


N
100%
TCCC Agent Demo
sip:xxxUserId@1400xx
Log in
quit
sip:13430xxxxx@1400xxxx.tccc.qcloud
call
Answer
hang up
You have a new call
1343 incoming call, please answer it
Reject
Answe

iOS

Last updated：2025-01-09 15:23:21

iOS Demo Quick Run

Tencent Cloud Contact Center iOS SDK enables agents to handle calls via PC, SIP phones, or mobile.
This guide shows how to quickly set up and run the iOS demo.

Developer Environment Requirements

Xcode 9.0+.
iPhone/iPad with iOS 9.0+: Device must run iOS 9.0 or later.
Valid developer signing: Proper developer certificates and provisioning profiles must be set up for testing on a real device.

Prerequisites

Login Tencent Cloud Console.
You have activated Cloud Contact Center application.
Completed Connecting Your Own Number and finished IVR Management.

Key Concepts

1. **SdkAppId**: A unique application ID created in Cloud Contact Center console. Each Tencent Cloud account can create up to 20 TCCC applications.
2. **UserID**: The member ID added in the Tencent Cloud Contact Center, usually an email address. For first-time setup, the admin account and password can be found in the Internal Message (sub-accounts must subscribe to TCCC product notification). Each SDKAppID supports multiple UserIDs, and additional licenses can be purchased at Agent Purchase.
3. **SecretId and SecretKey**: Developers need credentials to call the Cloud API. SecretId and Secretkey can be created in the Tencent Cloud console.
4. **Token**: A login credential obtained via CreateSDKLoginToken API. The app requests real-time tokens from your server, which stores the generation code and encryption key.

Steps

Step 1: Download Souce Code 'tccc-agent-ios-example'

Souce Code: tccc-agent-ios-example.

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

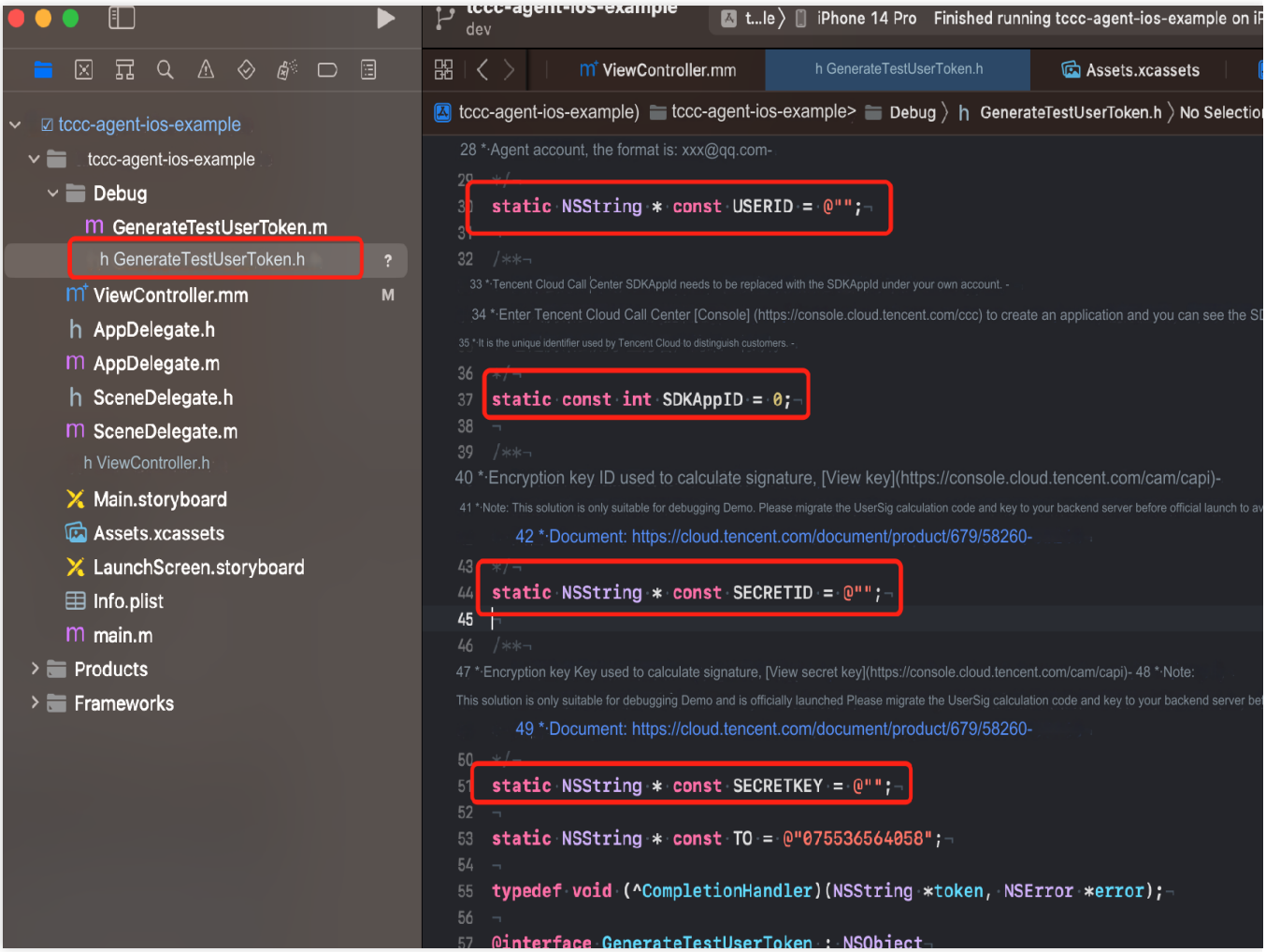
1. Open debug/GenerateTestUserToken.h file.
2. Set parameters in the GenerateTestUserToken.h file:

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



Caution:

Do not include the token generation code in your app's production version for the following reasons

The provided code is for testing basic SDK features only and is not secure for production. SECRETKEY in client-side code can be easily reverse-engineered, and web code is even more vulnerable. If your key is leaked, attackers can generate valid tokens and misuse your Tencent Cloud resources.
Safer approach is to place the token generation code and encryption key on your server. The app should request real-time tokens from your server as needed. This method is more secure, as breaching a server is significantly harder than cracking a client-side app. For more details, see Creating SDK Sign in Token

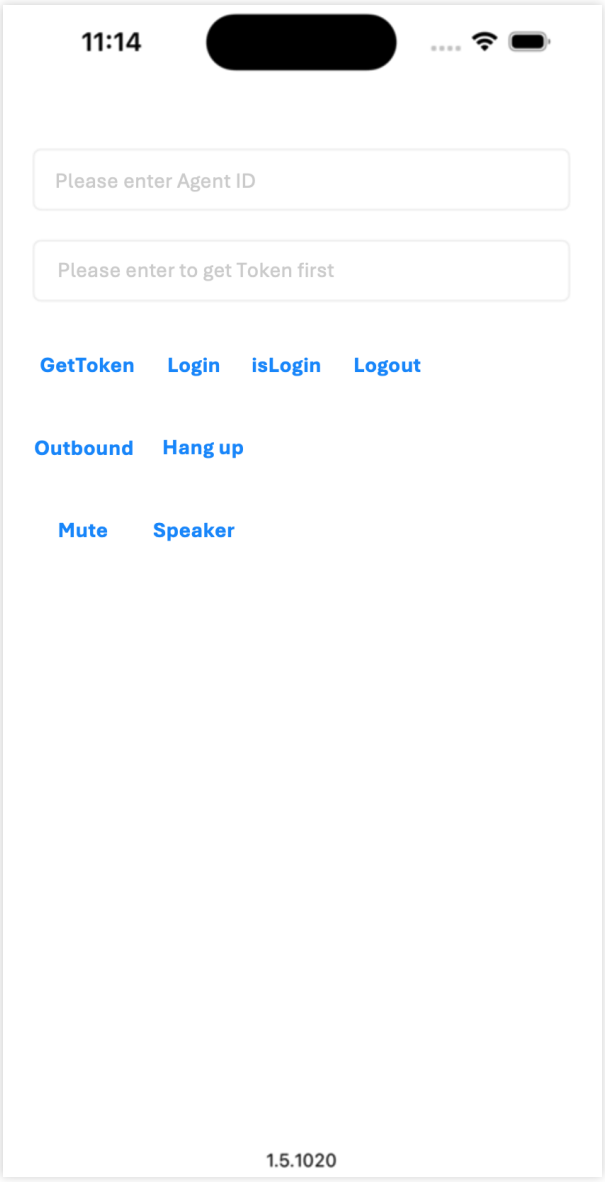
Step 3: Compile and Run the Demo

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

1. Click **Obtain Token** > **Log In**.
2. Click **Outbound Call** and start a call.

Showcase

The basic features are shown in the figure below:


11:14
Please enter Agent ID
Please enter to get Token first
GetToken
Login
isLogin
Logout
Outbound
Hang up
Mute
Speaker
1.5.1020

uni-app

Last updated：2025-01-09 15:23:21

This article introduces how to quickly run Cloud Contact Center uni-app Demo.

Development Environment Requirements

It is recommended to use the latest HBuilderX editor.
For iOS devices, iOS 9.0 or later is required, and the device must support audio.
For Android devices, version 4.1 or later is required, with audio support. Emulators are not currently supported. Ensure the "Enable Debugging" option is turned on.
Both iOS/Android devices must be connected to the Internet.

Prerequisites

Login Tencent Cloud Console
You have activated Cloud Contact Center application
Completed Bring Your Own Carrier via SIP Trunk and finished IVR Management.

Key Concepts

SDKAppID: A unique application ID created in Cloud Contact Center console. Each Tencent Cloud account can create up to 20 TCCC applications.
UserID: The member ID added in the Tencent Cloud Contact Center, usually an email address. For first-time setup, the admin account and password can be found in the Internal Message (sub-accounts must subscribe to TCCC product notification). Each SDKAppID supports multiple UserIDs, and additional licenses can be purchased at Agent Purchase if needed.
SecretId and SecretKey: Developers need credentials to call the Cloud API. SecretId and Secretkey can be created in theTencent Cloud console.
Token: A login credential obtained via CreateSDKLoginToken API. The app requests real-time tokens from your server, which stores the generation code and encryption key.

Steps

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

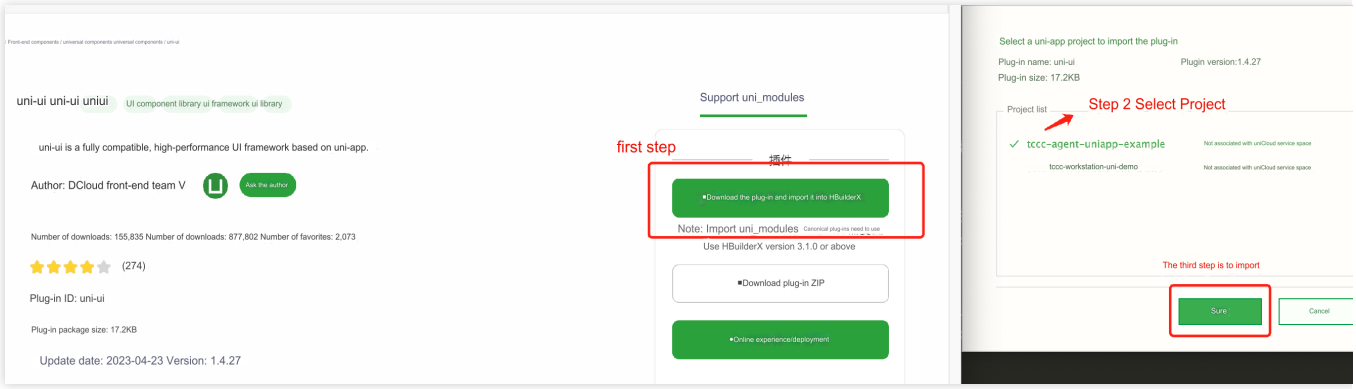
Source code: tccc-agent-uniapp-example

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 Project File 'tccc-agent-uniapp-example'

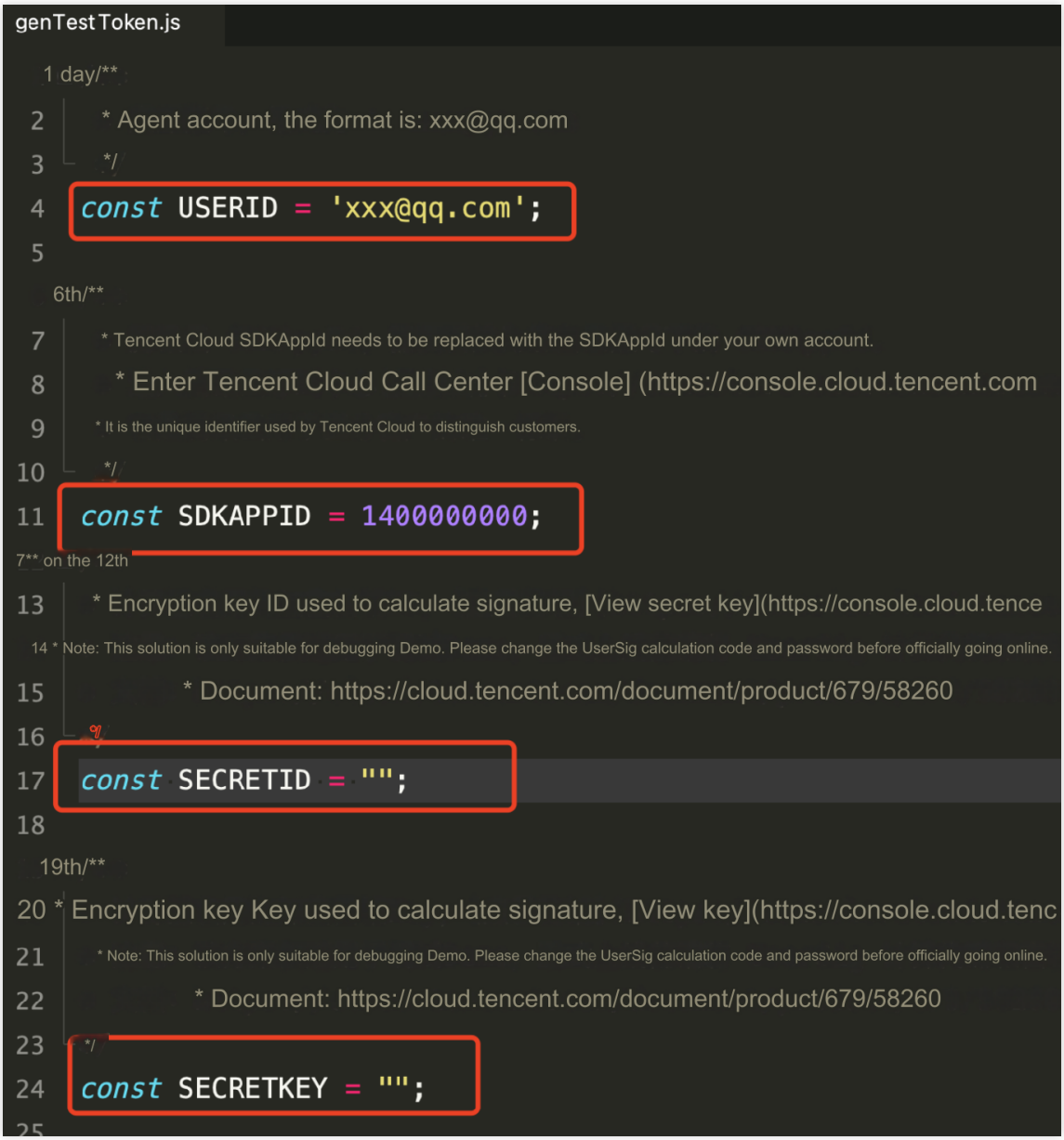
1. Open debug/genTestToken.js file.
2. Set parameters in genTestToken.js file:

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

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

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

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



Caution:

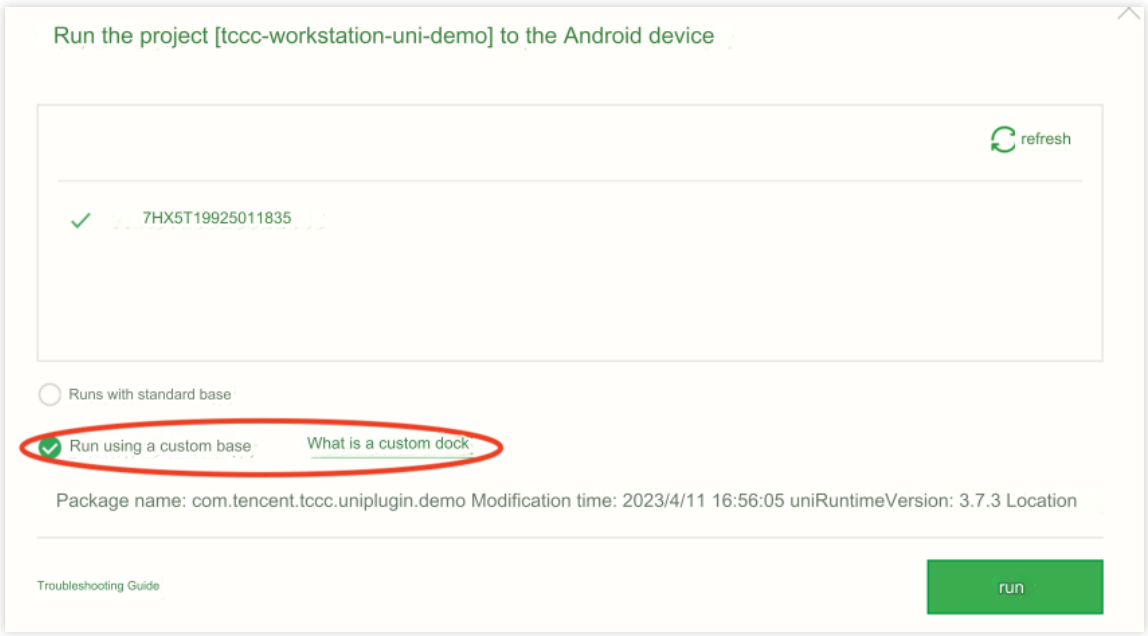
Do not include the token generation code in your app's production version for the following reasons:

The provided code is for testing basic SDK features only and is not secure for production. SECRETKEY in client-side code can be easily reverse-engineered, and web code is even more vulnerable. If your key is leaked, attackers can generate valid tokens and misuse your Tencent Cloud resources.

Safer approach is to place the token generation code and encryption key on your server. The app should request real-time tokens from your server as needed. This method is more secure, as breaching a server is significantly harder than cracking a client-side app. For more details, see Creating SDK Login Token.

Step 4: Compile the Demo

Use a custom base for packaging and running (do not select the standard base), and make sure to run the custom base on a physical device.



Caution:

For details on what a custom debugging base is and how to use it, please see the official tutorial.

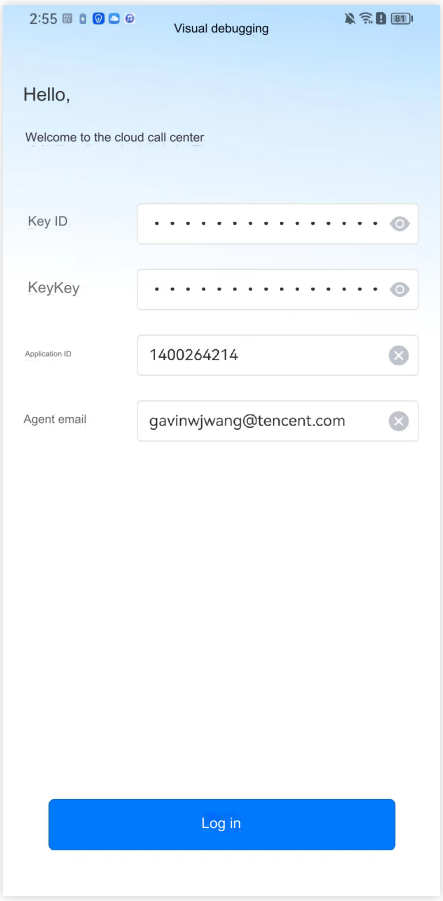
Step 5: Run the Demo

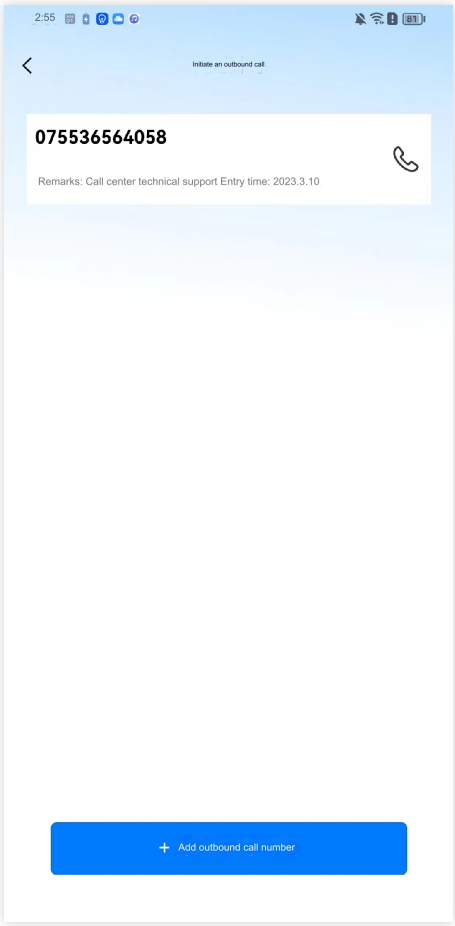
1. After choosing to run on the **physical device**, click **Log In**.
2. After logging in successfully, enter the phone number and start a call.

Showcase

Basic features as shown below:

Login Page	Number Management Page	Dial Pa


2:55
Visual debugging
Hello,
Welcome to the cloud call center
Key ID
KeyKey
Application ID
1400264214
Agent email
gavinwjwang@tencent.com
Log in

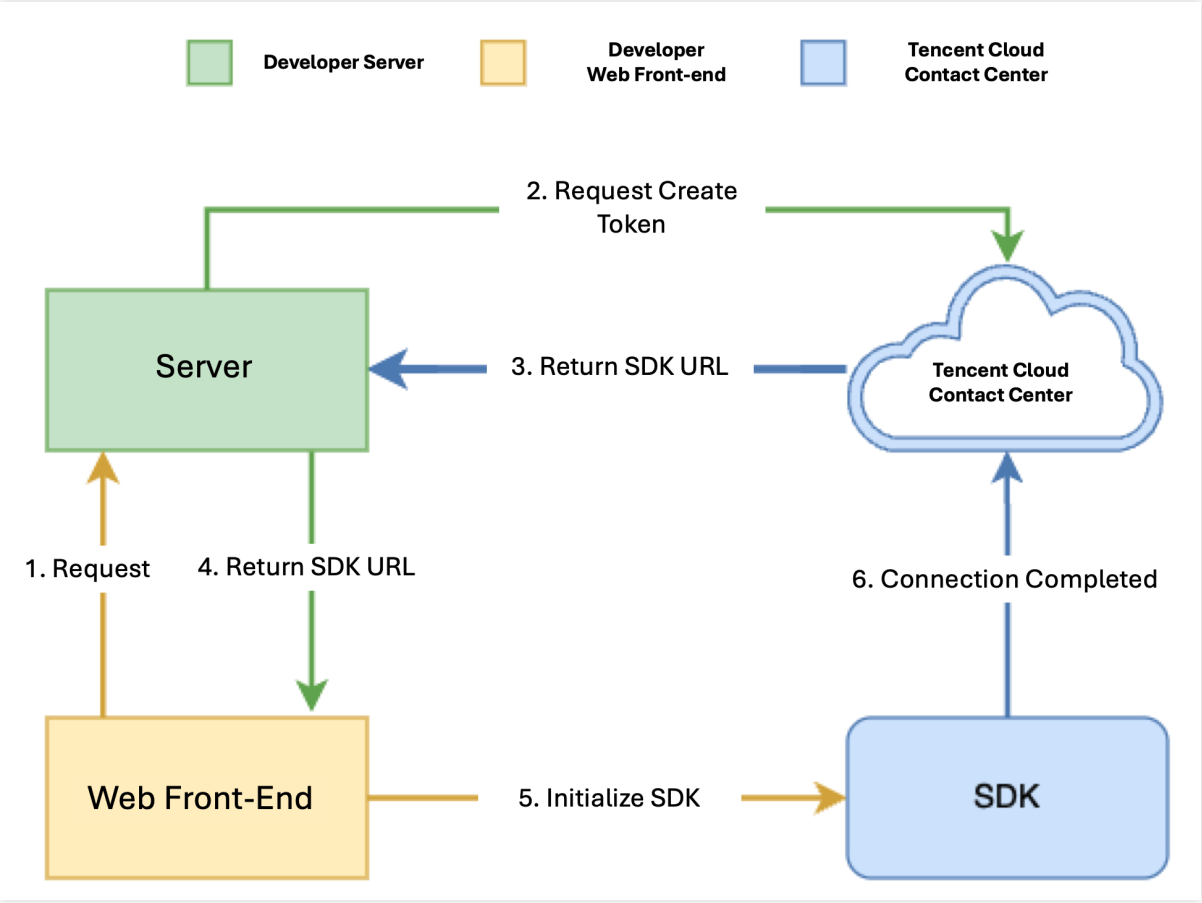

2:55
Initiate an outbound call
075536564058
Remarks: Call center technical support Entry time: 2023.3.10
Add outbound call number

Initializes SDK
Web

Last updated：2025-02-13 10:21:00

Flow

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. TCCC Web SDK supports Chrome 56+ and Edge 80+. It’s recommended to use the latest version for more features.

2. Use HTTPS for deploying frontend pages (localhost is fine for development); otherwise, calls may not work due to browser restrictions.

Integration Prerequisites

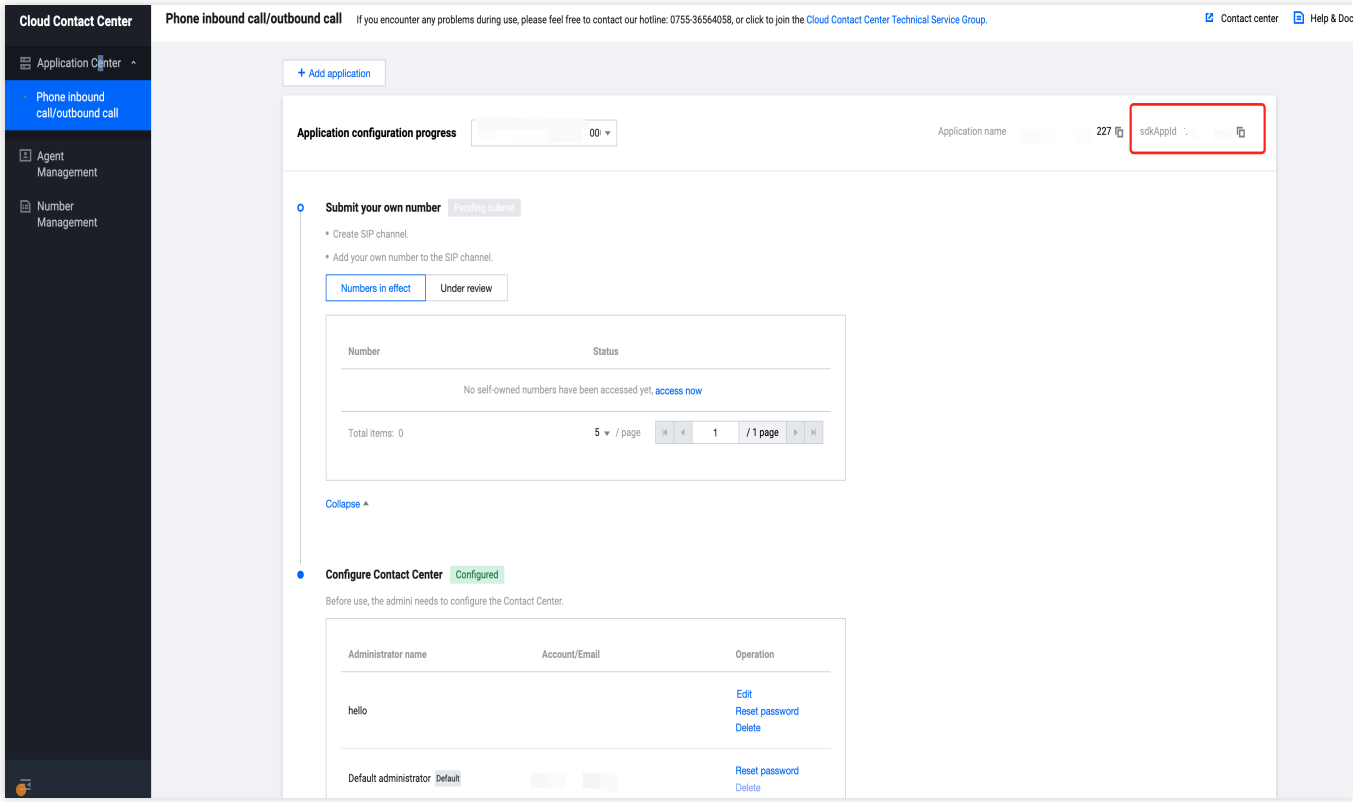
1. Cloud Contact Center application created.
2. You have purchased and added agent account.

Key Concepts

SdkAppId: A unique application ID created in Tencent Cloud Console. Each Tencent Cloud account can create up to 20 TCCC applications.
UserID: The member ID added in the Tencent Cloud Contact Center, usually an email address. For first-time setup, the admin account and password can be found in the Internal Message (sub-accounts must subscribe to TCCC product notification). Each SDKAppID supports multiple UserIDs, and additional licenses can be purchased at Agent Purchase.
SecretId and SecretKey: Developers need credentials to call the Cloud API. SecretId and Secretkey can be created in the Tencent Cloud console.
SDKURL: JS URL Created via the cloud API, valid for 10 minutes. Use it only once for SDK initialization. Request URL each time when initialize the SDK. Once initialized, no need to re-create.
SessionId: A unique ID for the entire call, from start to finish, whether it's the user's or the agent's call. It can be used to retrieve recordings, call logs, events, and more.

Step 1: Get Required Parameters

1. Get Tencent Cloud Contact Center `SecretId` and `SecretKey` . For guide, see Access Key.
2. Get TCCC application SDKAppID, and log into Tencent Cloud Console to view:



Step 2: Get SDK URL

Note: This step requires implementation on the integrator's backend.

1. Integrate tencentcloud-sdk
2. Call CreateSDKLoginToken API.
3. Return the SDK URL to the frontend.

The step uses `/loginTCCC` API. The example code is in Node.js; for 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: 'ap-singapore',
    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 the SDK URL On the Web Frontend and Complete the Initialization

Note: This step requires implementation on the integrator's frontend.

1. Request `/loginTCCC` API implemented in step two to get SDKURL.

2. Insert SdkURL into the page using script method.

3. Monitor tccc.events.ready event, and upon success, execute your business flow 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 direc
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);
  })
```

Android

Last updated：2025-01-09 15:25:53

Android SDK Quick Run

Tencent Cloud Contact Center Android SDK enables agents to handle calls via PC, SIP phones, or mobile.
This guide shows how to quickly set up and run the Android SDK.

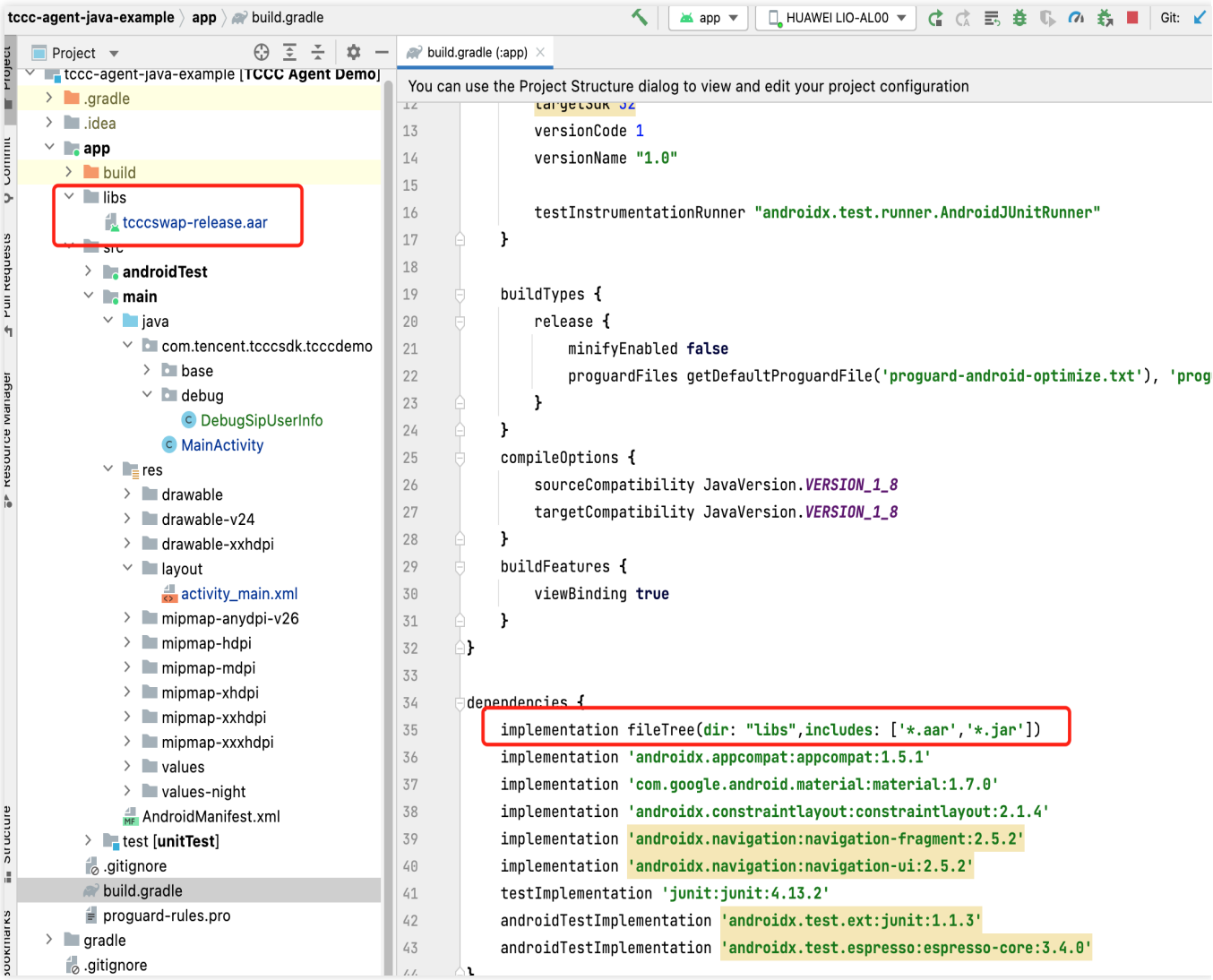
Developer Environment Requirements

Android Studio 3.5+.
Android 4.1 (SDK API 16) or later.

Integrating SDK (aar, jar)

Manual Download (aar, jar)

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




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

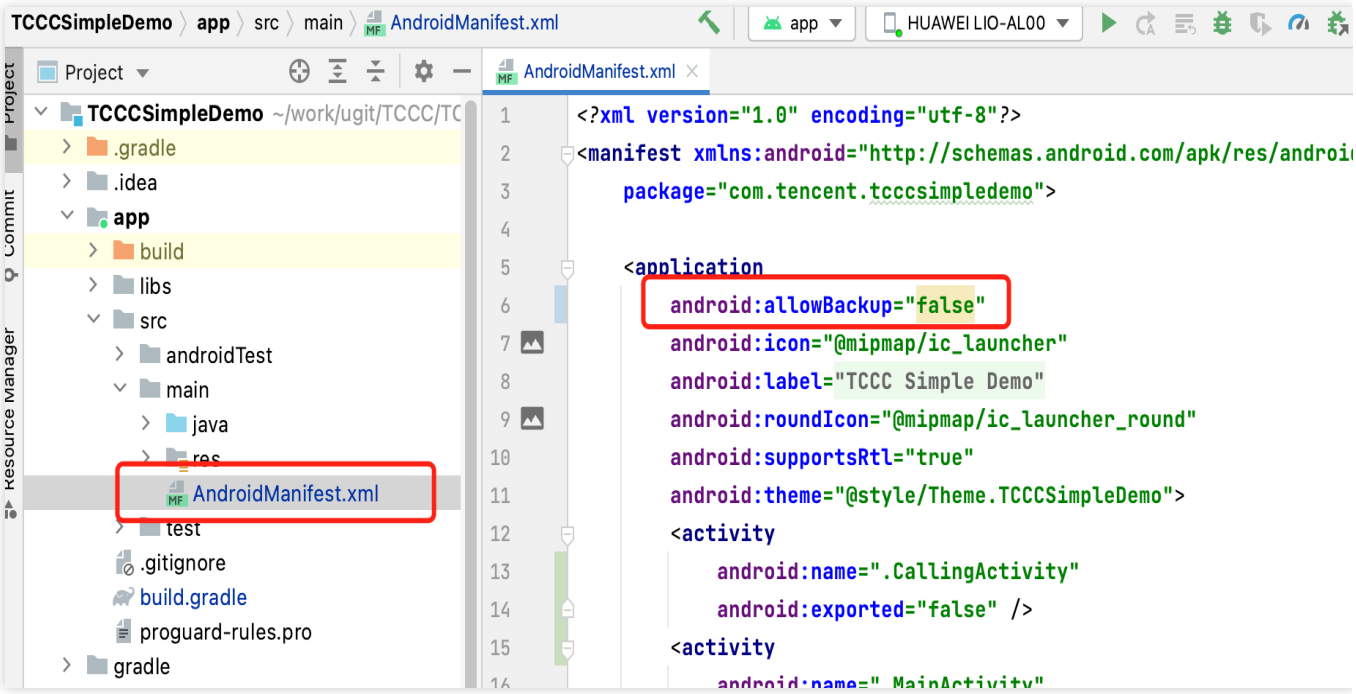
4. Specify the CPU architecture used by the app in defaultConfig in app/build.gradle.

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

Note:

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

5. In `app/src/AndroidManifest.xml` , change allowBack equal to false and restore infrastructure.



6. Click

Sync Now to complete Agent SDK integration.

Configure APP Permission

In AndroidManifest.xml, 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" />
```

Setup 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：2025-01-09 15:25:53

iOS Agent SDK Quick Run

Tencent Cloud Contact Center iOS Agent SDK enables agents to handle calls via PC, SIP phones, or mobile.
This guide shows how to quickly set up and run the iOS Agent 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: Use 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/CppSDKRelease/TCCCSDK_Ios.podspec'
  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
```

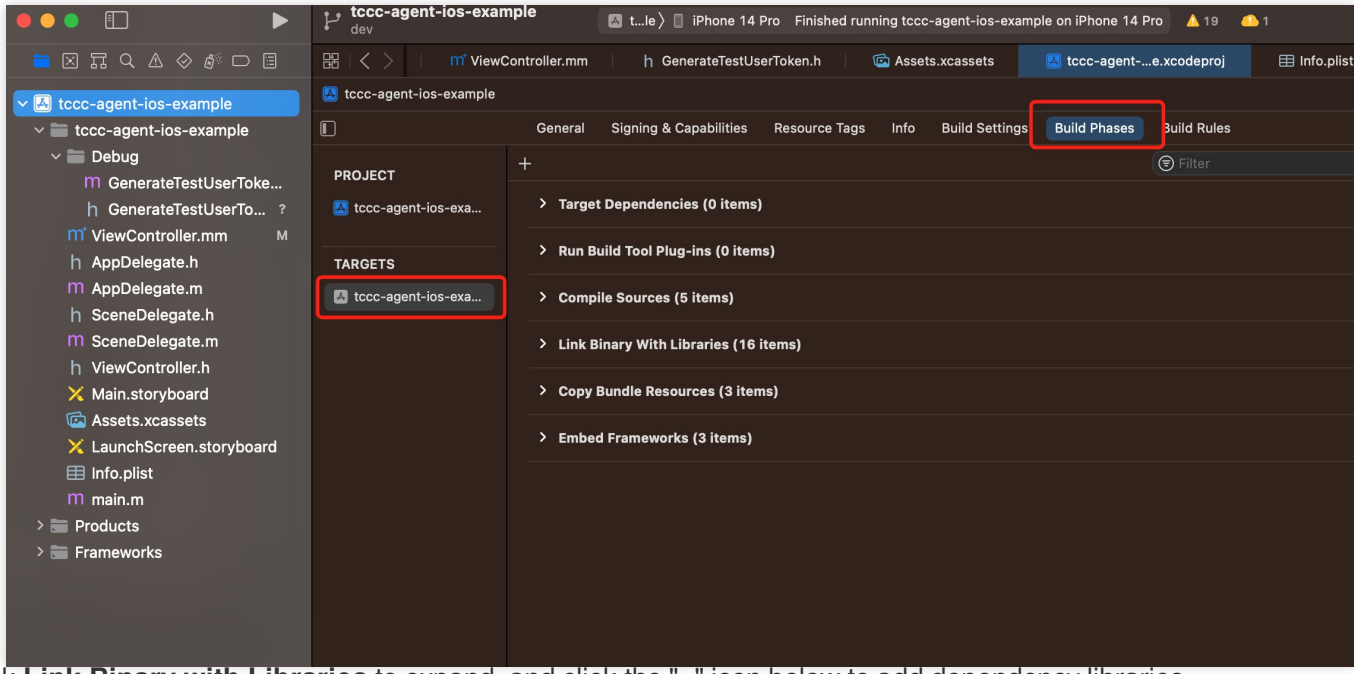
Alternatively, run the following command to update the local repository:

```
pod update
```

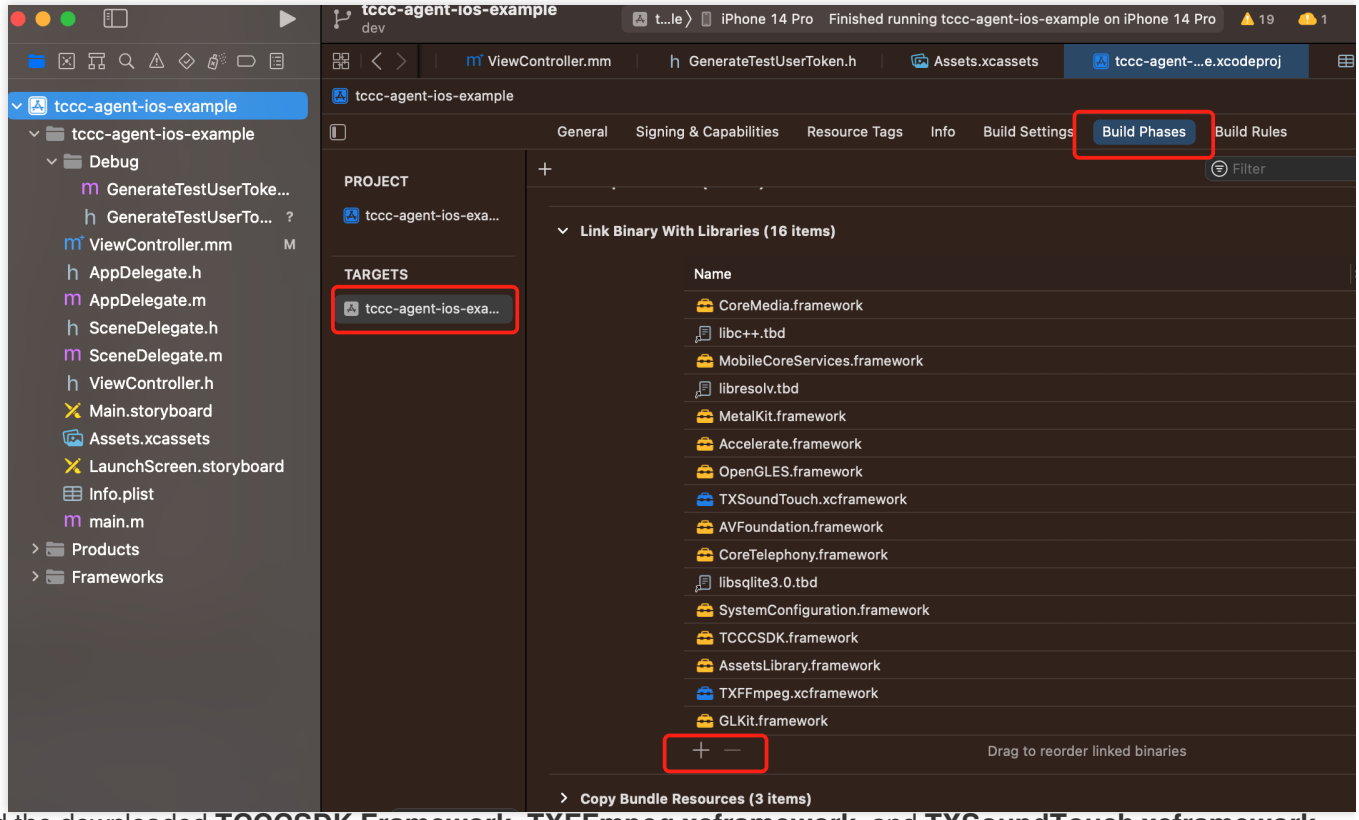
An XCWORKSPACE project file integrated with the SDK will be generated. Double-click to open it.

Solution 2: Download TCCC Agent SDK

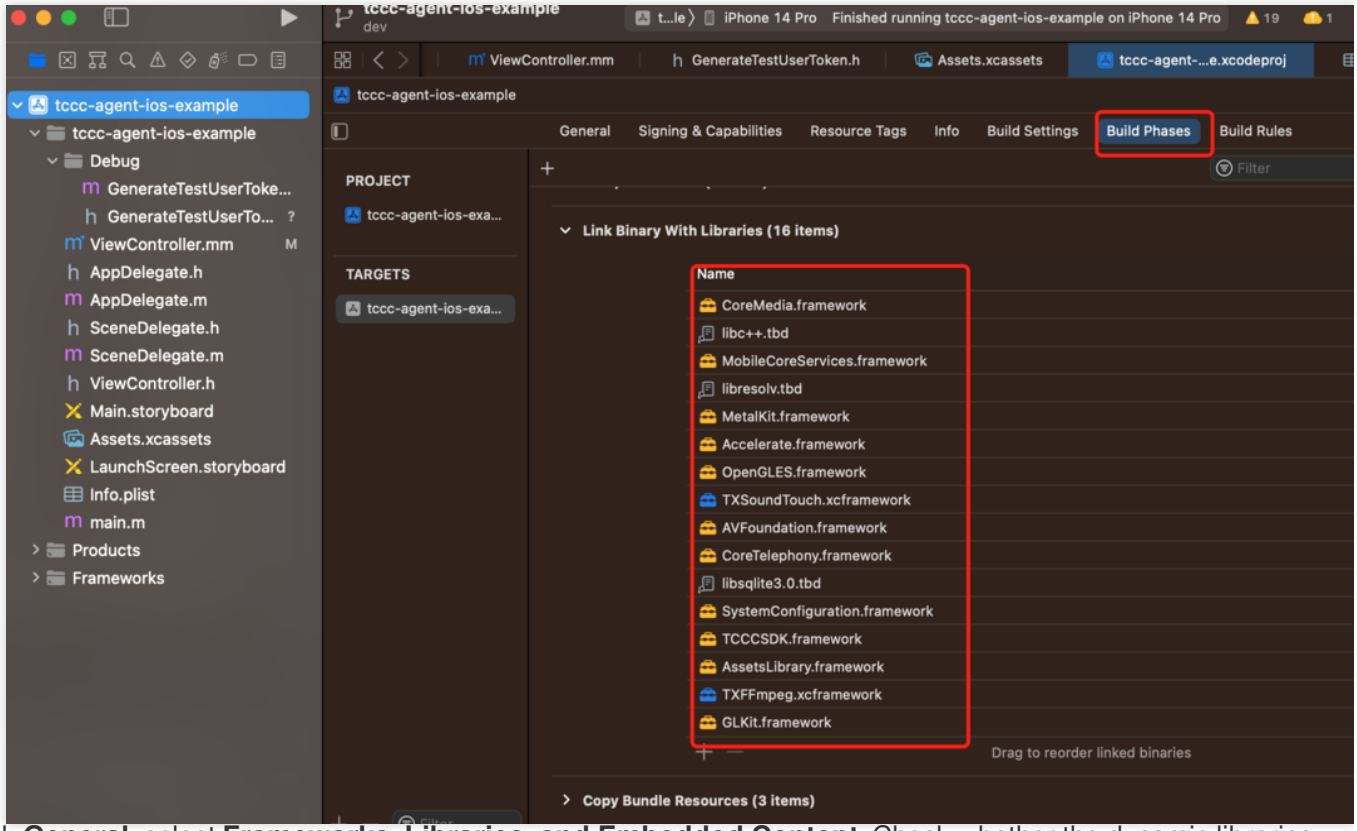
1. Download latest TCCC Agent SDK.
2. Open your Xcode project, select the target to run, and click **Build Phases**.



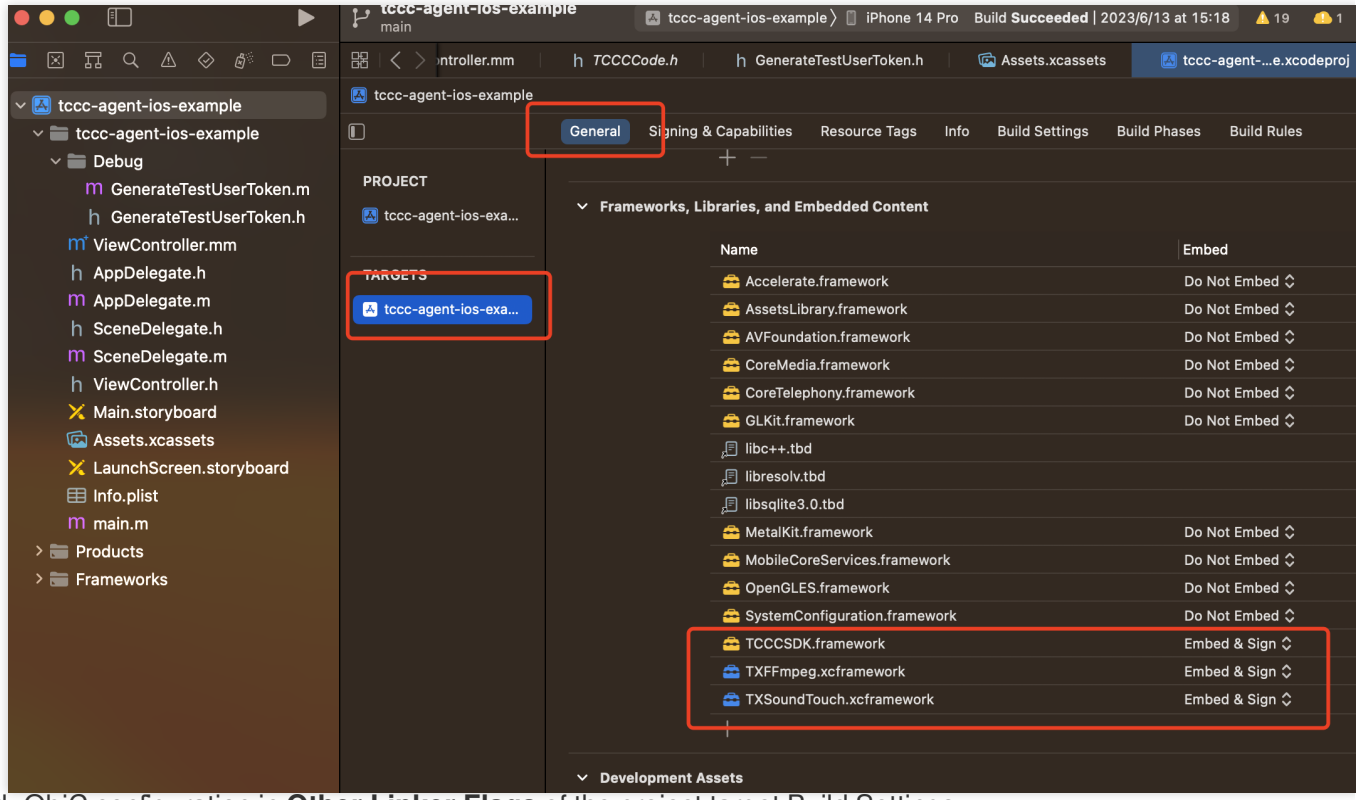
3. Click **Link Binary with Libraries** to expand, and click the "**+**" icon below to add dependency libraries.



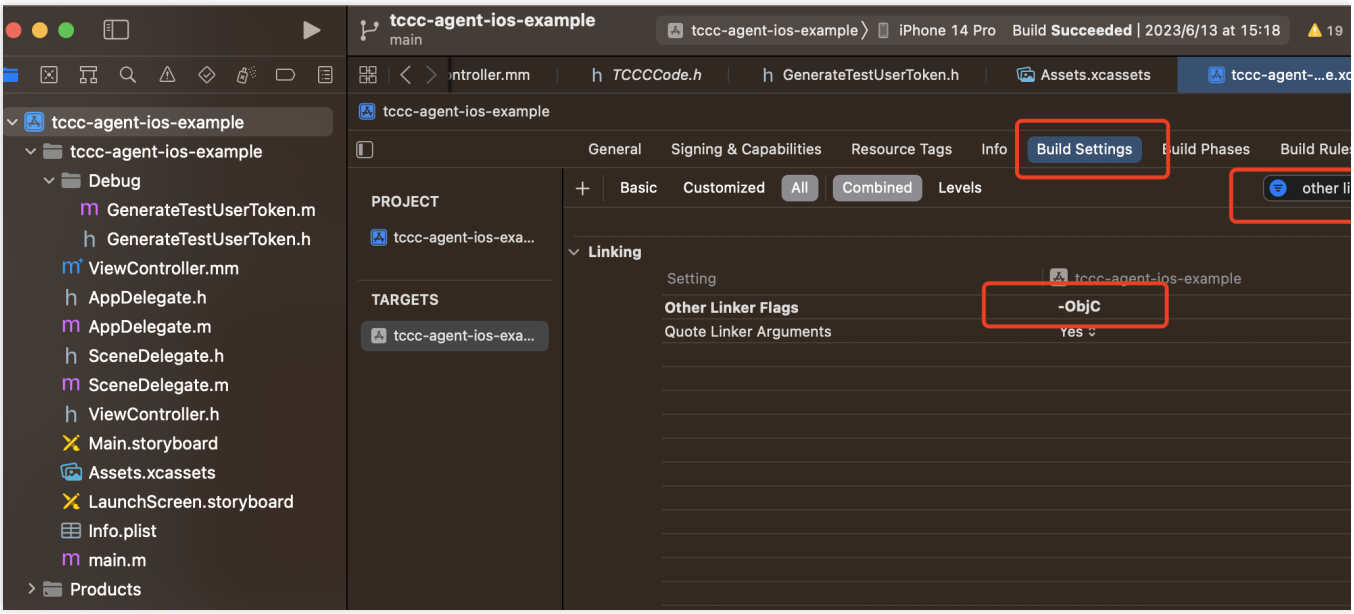
4. Add the downloaded **TCCCSDK.Framework**, **TXFFmpeg.xcframework**, and **TXSoundTouch.xcframework,** and the required dependency libraries **GLKit.framework**, **AssetsLibrary.framework**, **SystemConfiguration.framework**, **libsqlite3.0.tbd**, **CoreTelephony.framework**, **AVFoundation.framework**, **OpenGLES.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 **TCCCSDK.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.

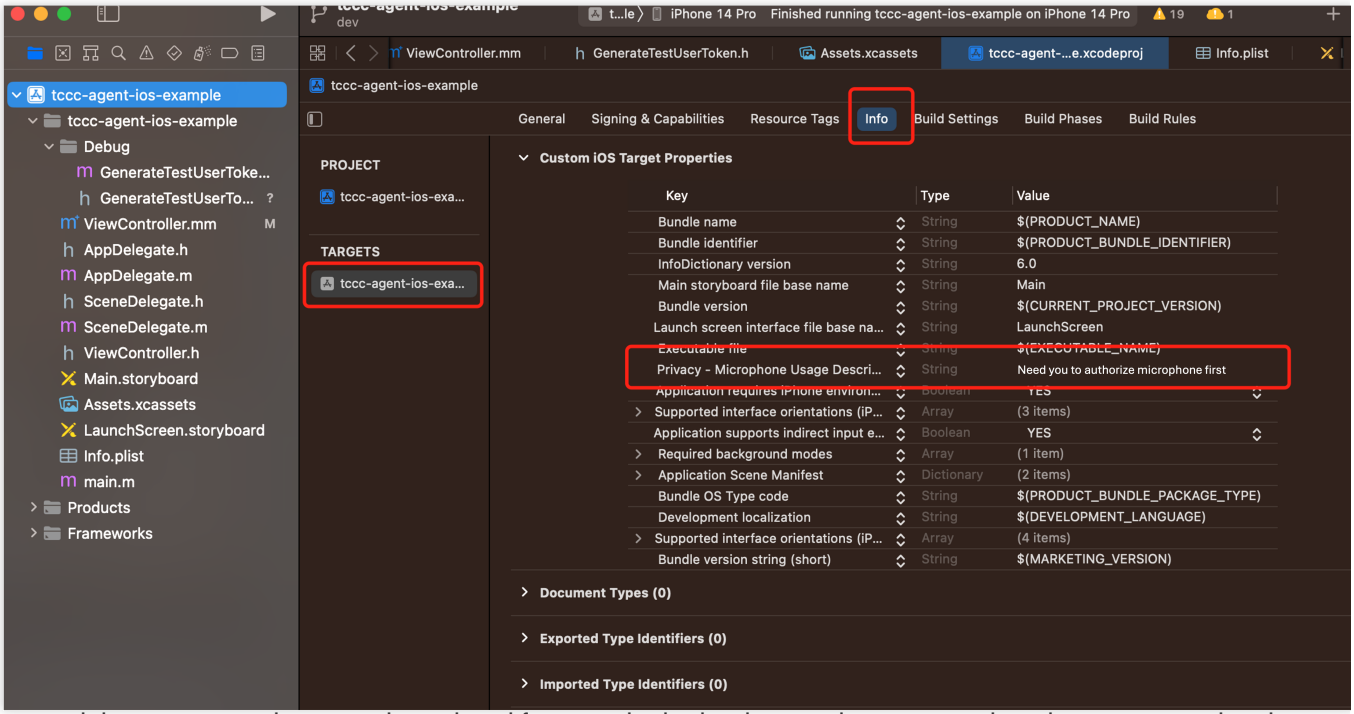


6. Add -ObjC configuration in **Other Linker Flags** of the project target Build Settings.

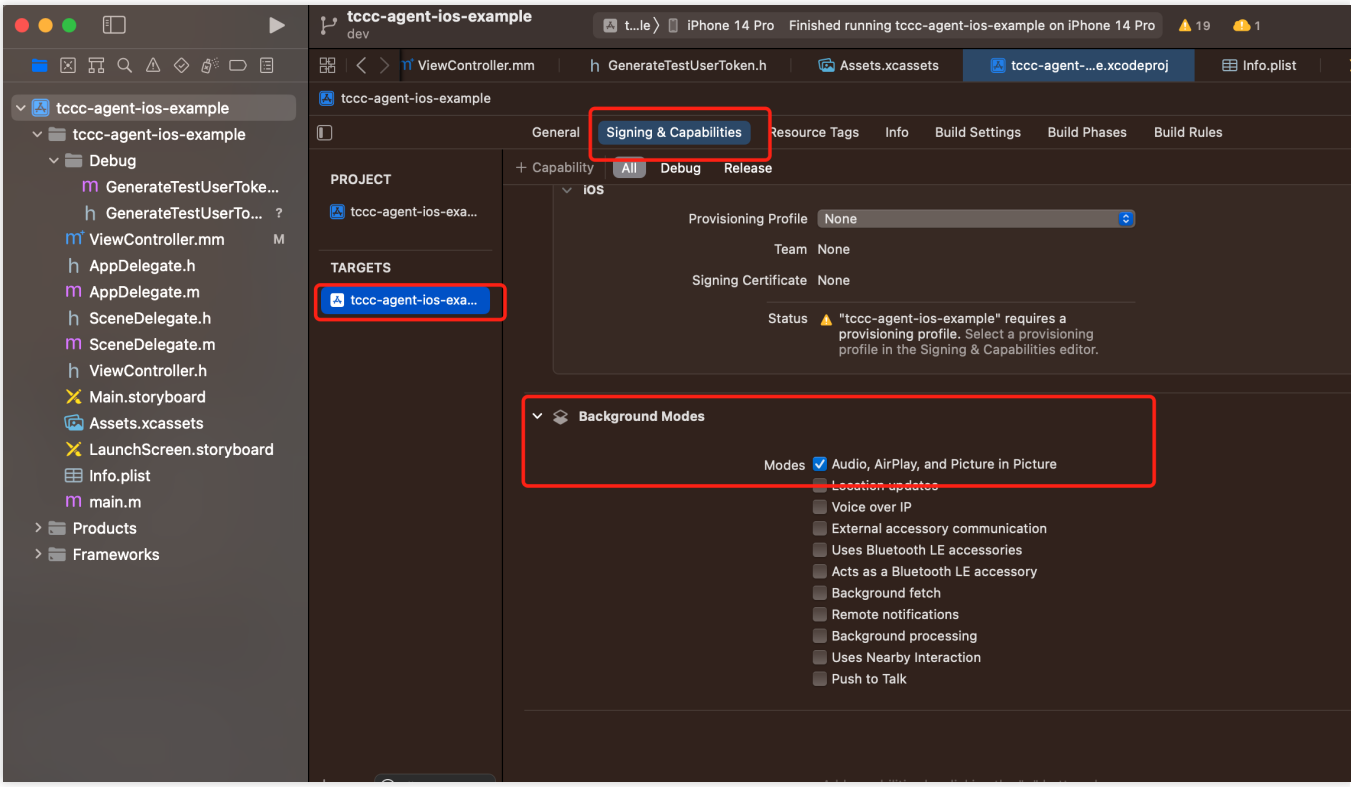


Configuring Permissions

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 "TCCCSDK/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.
});
```

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

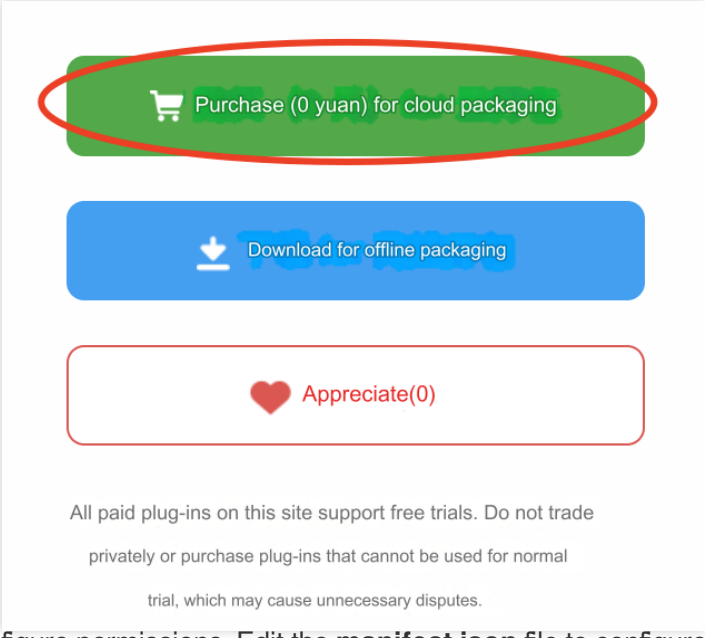
1. **SdkAppId**: The application ID users create on the Cloud Contact Center console. You can create up to 20 Cloud Contact Center applications under one Tencent Cloud account, often starting with 140.
2. **UserID**: The account configured by the agent or administrator in the Cloud Contact Center, usually 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**: 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.

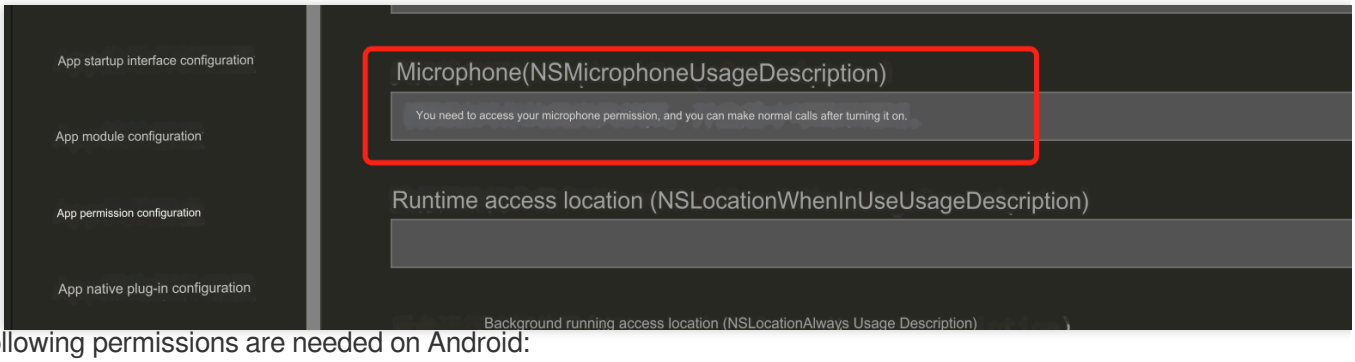
```
npm i tccc-sdk-uniapp
```

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.

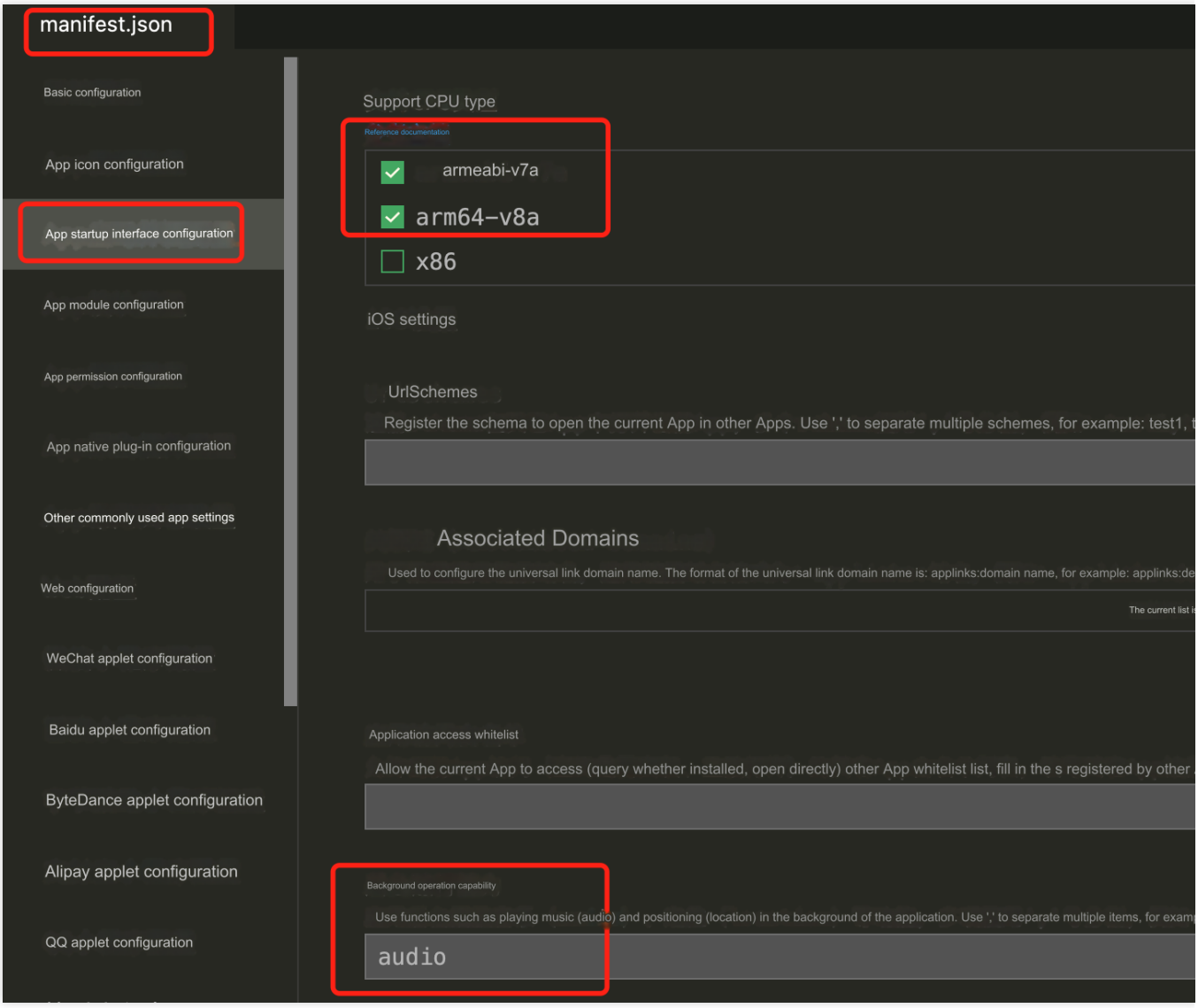


The following permissions are needed on Android:

```
<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" />
```

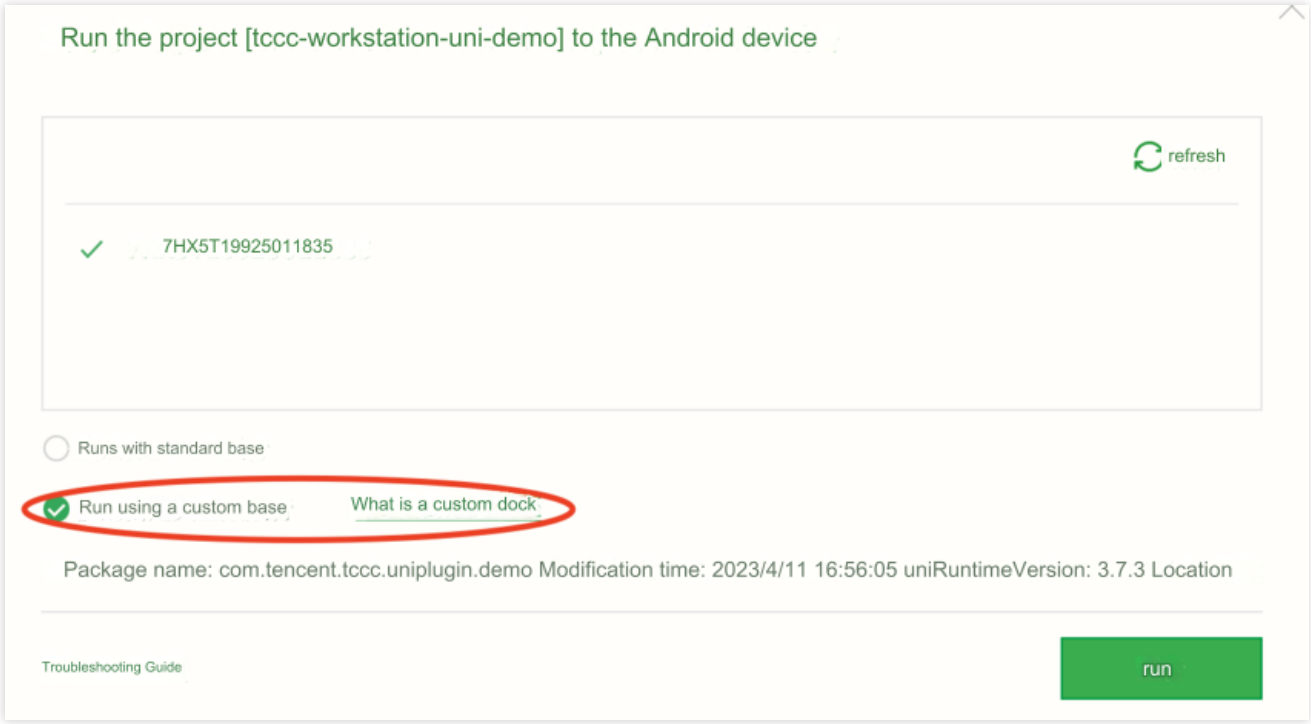
4. Configure audio to run in the background. When the mobile application is switched to the background, the operating system will pause the application's process to conserve resources. This means that all activities of the application will be stopped, including audio playback. On iOS, you need to configure **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.



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();
```

Workstation SDK: API Guide
Web

Last updated：2025-01-09 15:48:27

Note

TCCC becomes a global variable after the SDK is loaded, allowing direct access to its functionalities.

Data Structure

AgentStatus

Agent status.

Field	Description
free	Idle: Agent is online at their workstation but not currently engaged in a call.
busy	In Call: Agent is handling an inbound or outbound call and will not receive new call assignments during this time.
arrange	After-Call Work (ACW): Admins can set a wrap-up timer for agents to organize notes, during which no new calls are assigned.
notReady	Busy: agent will not receive new inbound calls, but can make outbound calls.
rest	On Break: agents can select a break reason such as "Meal," "Meeting," or "Training" (reasons are configured by the admin in the Management Panel).

ServerType

Service Endpoint Type: Describes the type of agent device used during call sessions.

Field	Description
staffSeat	Web agent to make and answer calls through the web server.
staffPhoneSeat	Agent who make and answer calls through

	mobile device.
miniProgramSeat	Agent who make and answer calls through wechat mini-program.
staffExtensionSeat	Agent who make and answer calls through SIP phone device.

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 Function of Voice and Audio Agent)

Inbound call

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

Parameter		Type	Required	Remarks
options	phoneNumber	String	Yes	Callee 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 through call CDR data push data push after being input
	skillGroupId	String	No	Skill group ID associated with outbound call number.
	callerPhoneNumber	String	No	Number that uses to make outbound call.
	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	Callee number location
	calleePhoneNumber	String	Yes	Callee Number
	callerPhoneNumber	String	Yes	Caller Number that is used to make the outbound call
	serverType	String	Yes	Service Endpoint Type: Describes the type of agent device used during call sessions. Enum value: staffSeat, staffPhoneSeat, and staffExtensionSeat. For details, see Session Service Type.
	remark	String	No	Callee number remarks

Answer Video Session

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

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

End Video Session

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Delete Call

tccc.Call.deleteCall(options)

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Mute

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Unmute

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Whether Mic is Muted

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Initiate Internal Call

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

Parameter		Type	Required	Remarks
options	calleeUserId	String	Yes	Agent account (Callee Side)
	useMobile	Boolean	No	Whether to call agent mobile or not

Transfer Video 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 on Hold

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Call Monitoring

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

Start call monitoring with audio as the default option, allowing only one session at a time. If textOnly: true is enabled, you can monitor multiple text sessions simultaneously, but this requires real-time speech to text feature to be turned on.

Note: This API can only be called by users with an administrator or quality inspector role.

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	The monitored conversation ID can be obtained from the Get PSTN Session List.
	textOnly	Boolean	No	The default is false, which means audio monitoring is initiated. Set it to true to enable text monitoring instead.

Call Intercept

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

Admin can take over a in progress call during monitoring, replacing the current agent. When this happens, the original agent will automatically exit the call (Call Intercept can only be initiated during monitoring).

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session has been taken over (intercepted) by Admin.

Cancel Call on Hold

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Send Phone 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 (Desk Agent API Functions)

Answer Video Session

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

End Chat Session

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Transfer Video 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 Agent API Functions)

Answer Video Session

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

End Video Session

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Mute

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Unmute

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Camera Turns Off

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Camera Turns On

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

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID

Transfer Video 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.

Swtich to Online Status

tccc.Agent.online(): void

Swtich to Offline Status

tccc.Agent.offline(): void

Agent Status Setup

tccc.Agent.setStatus(optoins): Promise<CommonSDKResponse>

Parameter		Type	Required	Remarks

options	status	String	Yes	Agent status, valid values: free: Idle, available to start service rest: On break arrange: After-call-work notReady: Busy stopNotReady: Stops showing busy
	restReason	String	No	Break Reason

Get Agent Status

tccc.Agent.getStatus():AgentStatus

Devices (Device API Functions)

Web Browser Availability Check

tccc.Devices.isBrowserSupported(): boolean

Note:

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

Return Microphone List

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

Return Speaker List

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

UI (User Interface API Functions)

Hide All SDK UIs

tccc.UI.hide(): void

Show All SDK UIs

tccc.UI.show(): void

Show Floating Button

tccc.UI.showfloatButton(): void

Hide Floating Button

tccc.UI.hidefloatButton(): void

Show Workstation

tccc.UI.showWorkbench(): void

Hide Workstation

tccc.UI.hideWorkbench(): void

Display Call Toolbar

tccc.UI.showNotificationBar(): void

Hide Call Toolbar

tccc.UI.hideNotificationBar(): void

Modify SDK Local Settings

You are able to turn off SDK ringtone and system notifications

tccc.UI.updateUserCustomSettings(settings): void

All parameters in settings are optional and support incremental updates.

Parameter		Type	Required	Remarks
settings	disableRingtone	Boolean	No	true means disable SDK's ringtones (e.g., inbound call and answer tones).
	disableNotification	Boolean	No	true means disable SDK system notifications

Events

Monitor Events

tccc.on(event, callback)

Cancel Event Monitor

tccc.off(event, callback)

Complete SDK Initialization

tccc.events.ready

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

Callback Parameter		Type	Required	Remarks
options	tabUUID	String	Yes	A unique ID for the current page that changes after refresh, used for multi-tab SDK integration.

Inbound Session

tccc.events.callIn

Types of inbound sessions include:
phone: voice session (Phone)
im: chat session (Desk)
voip: Audio session
video: Video session
internal: Internal session

Inbound Voice Call

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	type	'phone'	Yes	Type of voice session (e.g., Web, mobile, sip phone calls)
	timeout	Number	Yes	Session connection timeout duration, (0 means no timeout is applied)
	calleePhoneNumber	String	Yes	Callee Number
	callerPhoneNumber	String	No	Caller Number
	callerLocation	String	No	Caller number location
	remark	String	No	Remarks
	ivrPath	{key: String, label: String}[]	-	User IVR path: KEY represents the key pressed at each node, and LABEL

				represents the label for the key.
	protectedCallee	String	No	When admin enables number masking to protect real numbers, the masked number will be shown in this field for the callee.
	protectedCaller	String	No	When admin enables number masking to protect real numbers, the masked number will be shown in this field for the caller.
	serverType	'staffSeat' | 'staffPhoneSeat' | 'staffExtensionSeat'	Yes	Indicates the device used by the agent to answer incoming calls: 1) staffSeat (default): Web agent. 2) StaffPhoneSeat: Agent's mobile phone. 3) MiniProgramSeat: Mini Program agent. 4) staffExtensionSeat: Agent's linked desk phone.

Inbound Chat Session

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	type	'phone'	Yes	Type of voice session (e.g., Web, mobile, sip phone calls)
	timeout	Number	Yes	Session connection timeout duration, (0 means no timeout is applied)
	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

Inbound Audio Session (VoIP Call)

Parameter	Type	Required	Remarks

options	sessionId	String	Yes	Session ID
	type	'voip'	Yes	Audio session type
	timeout	Number	Yes	Session connection timeout duration, (0 means no timeout is applied)
	callee	String	Yes	Channel
	calleeRemark	String	No	Channel 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 number location
	ivrPath	{key: String, label: String}[]	No	User IVR path: KEY represents the key pressed at each node, and LABEL represents the label for the key.
	clientData	String	No	User-defined parameter

Inbound Video Session

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	type	'video'	Yes	Video session type
	timeout	String	Yes	Session connection timeout duration, (0 means no timeout is applied)
	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

Inbound Internal Session

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	type	'internal'	Yes	Internal session type
	timeout	Number	Yes	Session connection timeout duration, (0 means no timeout is applied)
	peerUserId	String	Yes	Caller's account

Agent Session Handling

tccc.events.userAccessed

Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	tabUUID	String	No	Present in multi-tab integration, it indicates which tab the agent used to answer.

Session Transfer Timeout 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	This is only applied in voice call, present when the disconnector is "admin," indicating the reason for the hung up.
	subReason	String	No	This is only applied in voice call, present when the disconnector is "admin," indicating the detailed reason for the hung up.

Outbound Call Succeeded Event

tccc.events.callOuted

Callback Parameter		Type	Required	Remarks
options	sessionId	String	Yes	Session ID
	callerPhoneNumber	String	Yes	Caller number used for making an outbound call
	calleePhoneNumber	String	Yes	Callee Number
	serverType	'staffSeat' | 'staffPhoneSeat' | 'staffExtensionSeat' | 'MiniProgramSeat'	Yes	Indicates agent uses what device to make outbound calls: `staffSeat` - default value, indicating agent is using web browser to make calls. StaffPhoneSeat indicates using mobile outbound call MiniProgramSeat indicates using mini program outbound call staffExtensionSeat indicates using the phone for outbound calls
	tabUUID	String	No	Once multi-tab integration turned on, it indicates which tab the agent used to answer.

Outbound Call Answered 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.	

Agent Being Kicked Out Event

tccc.events.kickedOut

Triggered When Agent Logs in on Multiple Devices

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 results. For more structure info, see ASR Document.	
	flow	'IN' | 'OUT'	Yes	Recognition direction IN: User side OUT: Agent side	

Multi-Tab Integration SDK

By default, TCCC Web SDK only allows to login device. Multiple logins will trigger the **kickedOut** event. After enabling the multi-Tab feature, calls initiated from any page will be displayed on other pages. Developers can hide the

UI according to business logic or monitor to corresponding events for handling.

Restriction

1. One browser mutli-window, no incognito mode.
2. The SDK is integrated within the business system under the **same domain name**.
3. Mobile browsers are not supported.

Integration Guidance

1. Initialize the SDK, refer to Web.
2. Add the **enableShared** parameter to enable multi-tab feature.

```
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');
      script.src = SdkURL;
      /*
      * Add enableShared to enable multi-tab feature
      */
      script.dataset.enableShared = 'true'
      document.body.appendChild(script);
      script.addEventListener('load', () => {
        window.tccc.on(window.tccc.events.ready, ({ tabUUID }) => {
          resolve('Initialization succeeded, current tabUUID is ' + tabUUID)
        });
        window.tccc.on(window.tccc.events.tokenExpired, ({message}) => {
          console.error('Initialization failed', message)
          reject(message)
        })
      })
    })
}
```

3. Rational of multi-tab handling.

When the **callOuted** (outbound call succeeded) and **userAccessed** (agent answered successfully) events are triggered, the tabUUID field will be added to indicate which page initiated the call/answer.

```
let curTabUUID = '';

window.tccc.on(window.tccc.events.ready, ({ tabUUID }) => {
    console.log('Initialization succeeded, current tabUUID is ' + tabUUID)
```

```
    curTabUUID = tabUUID;
});

window.tccc.on(window.tccc.events.callOuted, ({ sessionId, tabUUID }) => {
    if (tabUUID && tabUUID !== curTabUUID) {
      // Received outbound call success event from another page, business logic can
    }
})

window.tccc.on(window.tccc.events.userAccessed, ({ sessionId, tabUUID }) => {
    if (tabUUID && tabUUID !== curTabUUID) {
      // Received call answered event from another page, business logic can handle
      // This is sample code, this event will be ignored
      return;
    }
})
```

Android

Last updated：2025-01-09 15:41:51

Create Application and Event Callback

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
checkLogin	Checks whether the SDK is already logged in.
logout	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 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	Mute.
unmute	Unmute.

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 prompt volume.
getAudioPlayoutVolume	Obtains the remote audio prompt volume.
setAudioRoute	Sets audio routing.

Debug APIs

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

Sample Code for Get 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 Callbacks for Cloud Connection Status

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 triggers this event callback when it loses connection to the clou
        * For example, this can happen if a user enters an elevator during a call.
        * During reconnection, SDK triggers **onTryToReconnect**, and once the conn
        * Therefore, SDK transitions between these three connection-related events
        */
    @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(TCCCServerType 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(TCCCServerType 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 check the error message corresponding to -3301 in onError to identify the cause of the failure.
ERR_RTC_REQUEST_IP_TIMEOUT	-3307	Request for IP and Sig timed out. Make sure the network is working and UDP is allowed through the firewall.

ERR_RTC_CONNECT_SERVER_TIMEOUT	-3308	Connection timed out. Check if the network is disconnected or a VPN is enabled. You can also try switching to 4G.
ERR_RTC_ENTER_ROOM_REFUSED	-3340	Room entry request was denied. Check if you're repeatedly calling enterRoom for the same room ID.

iOS

Last updated：2025-01-09 15:41:52

This article mainly introduces common APIs for the agent side of the Cloud Contact Center (TCCC). On iOS, we provide Swift, Objective-C, and C++ APIs for developer choice. We recommend iOS developers use Swift for application development.

Creating Instances and Event Callbacks

API	Description
sharedInstance	Creates an ITCCCWorkstation instance (singleton).
destroySharedInstance	Destroys an ITCCCWorkstation instance (singleton).
addTcccListener	Adds an ITCCCWorkstation event callback.
removeTCCCListener	Removes an ITCCCWorkstation event callback.

Sample Code for Creating Instances and Setting Event Callbacks

Swift

Objective-C

C++

```
import TCCCSDK

let tcccSDK: TCCCWorkstation = {
    // Create an instance.
   return TCCCWorkstation.sharedInstance()
}()

// Set the TCCC event callback
tcccSDK.addTcccListener(self)

// Remove the TCCC event callback
tcccSDK.removeTCCCListener(self)
// Destroy the instance
TCCCWorkstation.destroySharedIntance()
```

```
// Import the OC header file
#import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"

@property (strong, nonatomic) TCCCWorkstation *tcccSDK;
```

```
- (TCCCWorkstation*)tcccSDK {
    if (!_tcccSDK) {
        // Create an instance.
        _tcccSDK = [TCCCWorkstation sharedInstance];
    }
    return _tcccSDK;
}
// Set the TCCC event callback
[self.tcccSDK addTcccListener:self];

// Remove the TCCC event callback
[self.tcccSDK removeTCCCListener:self];
// Destroy the instance
[TCCCWorkstation destroySharedIntance];
 _tcccSDK = nil;
```

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

    void onError(TCCCError 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-related APIs

API	Description

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

Login and logout sample code

Swift

Objective-C

C++

```
import TCCCSDK

let param = TXLoginParams()
// 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 S
// Token](https://cloud.tencent.com/document/product/679/49227)
param.token = "";
// Cloud Contact Center application ID, which usually starts with 1400
param.sdkAppId = 0;
// Set to agent mode
param.type =  .Agent;
// Login
tcccSDK.login(param) { info in
    // Login succeeded
} fail: { code, message in
    // Login failed
}

// Check login status
tcccSDK.checkLogin {
  // Logged in
} fail: { code, message in
  // Not logged in or kicked out
}

// Logout
tcccSDK.logout {
  // Exit succeeded
} fail: { code, message in
  // Exit exception
}


// Import the OC header file
```

```
#import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"

TXLoginParams *param = [[TXLoginParams alloc] init];
// 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 S
// Token](https://cloud.tencent.com/document/product/679/49227)
param.token = @"";
// Cloud Contact Center application ID, which usually starts with 1400
param.sdkAppId = 0;
// Set to agent mode
param.type =  Agent;
[self.tcccSDK login:param succ:^(TXLoginInfo * _Nonnull info) {
    // Login succeeded
} fail:^(int code, NSString * _Nonnull desc) {
    // Login failed
}];

// Check login status
[self.tcccSDK checkLogin:^{
    // Logged in
} fail:^(int code, NSString * _Nonnull desc) {
    // Not logged in or kicked out
}];

// Logout
[self.tcccSDK logout:^{
    // Exit succeeded
} fail:^(int code, NSString * _Nonnull desc) {
    // Exit exception
}];
```

```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;
// Callback class for login
class TCCCLoginCallbackImpl : public ITXValueCallback<TCCCLoginInfo> {
public:
    TCCCLoginCallbackImpl() {

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

    void OnError(TCCCError 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;
// Set to 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

Swift

Objective-C

C++

```
import TCCCSDK
```

```
let callParams = TXStartCallParams()
// Called phone number
callParams.to = "";
// Number remarks, which will replace the number displayed in the call bar (optiona
callParams.remark = "";
// Initiate an outbound call
tcccSDK.call(callParams) {
    // Call initiated successfully
} fail: { code, message in
    // Call initiation failed
}
// End the call
tcccSDK.terminate()
```

```
// Import the OC header file
#import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"

TXStartCallParams *callParams = [[TXStartCallParams alloc] init];
// Called phone number
callParams.to = TO;
// Number remarks, which will replace the number displayed in the call bar (optiona
callParams.remark = @"testByIos";
// Initiate an outbound call
[self.tcccSDK call:callParams succ:^{
    // Call initiated successfully
} fail:^(int code, NSString * _Nonnull desc) {
    // Call initiation failed
}];

// End the call
[self.tcccSDK terminate];
```

```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;
class TCCCCommonCallback : public ITXCallback {
private:
    NSString* mFunName;
public:
    TCCCCommonCallback(NSString* funName) {
        mFunName = funName;
    }
    ~TCCCCommonCallback() override {

    }
    void OnSuccess() override {
      // Succeeded
```

```
    }

    void OnError(TCCCError error_code, const char *error_message) override {
        std::string copyErrMsg = makeString(error_message);
      // Failed
    }
};
TCCCCommonCallback* startCallCallbackImpl = nullptr;
if (nullptr == startCallCallbackImpl) {
  startCallCallbackImpl = new TCCCCommonCallback(@"startCall");
}
TCCCStartCallParams 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.

Switching Audio Route Sample Code

Swift

Objective-C

C++

```
import TCCCSDK

// Switch to speaker
tcccSDK.getDeviceManager().setAudioRoute(.TCCCAudioRouteSpeakerphone)
// Mute
tcccSDK.mute()
// Unmute
```

```
tcccSDK.unmute()
```

```
// Import the OC header file
#import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"

// Switch to speaker
[[self.tcccSDK getDeviceManager] setAudioRoute:TCCCAudioRouteSpeakerphone];
// Mute
[self.tcccSDK mute];
// Unmute
[self.tcccSDK unmute];
```

```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;

// Switch to speaker
tcccSDK->getDeviceManager()->setAudioRoute(TCCCAudioRoute::TCCCAudioRouteSpeakerpho
// Mute
tcccSDK->mute();
// Unmute
tcccSDK->unmute();
```

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

Swift

Objective-C

C++

```
import TCCCSDK

// Obtain SDK version number
let version = TCCCWorkstation.getSDKVersion()
```

```
// Import the OC header file
#import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"

// Obtain SDK version number
NSString* version = [TCCCWorkstation getSDKVersion];
```

```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;

// 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

Swift

Objective-C

C++

```
import TCCCSDK

func onError(_ errCode: TCCCErrorCode, errMsg: String, extInfo: [AnyHashable : Any]
    // Error event callback
}
func onWarning(_ warningCode: TCCCCWarningCode, warningMsg: String, extInfo: [AnyHa
    // Warning event callback
}

// Set the TCCC event callback
tcccSDK.addTcccListener(self)
```

```
// Import the OC header file
#import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"

#pragma mark - TCCCDelegate
- (void)onError:(TCCCErrorCode)errCode errMsg:(NSString * _Nonnull)errMsg extInfo:(
    // Error event callback
}
```

```
- (void)onWarning:(TCCCCWarningCode)warningCode warningMsg:(NSString *_Nonnull)warn
    // Warning event callback
}

// Set the TCCC event callback
[self.tcccSDK addTcccListener:self];
```

```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;

// Set the callback. TCCCCallbackImpl needs to be derived from ITCCCCallback
class TCCCCallbackImpl:public ITCCCCallback {
public:
    TCCCCallbackImpl() {}
    ~TCCCCallbackImpl() {}
    // Error event callback
    void onError(TCCCError errCode, const char* errMsg, void* extraInfo) {}
    // Warning event callback
    void onWarning(TCCCCWarning warningCode, const char* warningMsg, void* extraInf
};
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

Swift

Objective-C

C++

```
import TCCCSDK

func onNewSession(_ info: TXSessionInfo) {
    // New session event, including inbound and outbound calls
```

```
}
func onAccepted(_ sessionId: String) {
    // Remote party answered event
}
func onEnded(_ reason: TXEndedReason, reasonMessage: String, sessionId: String) {
    // Call end event
}

// Set the TCCC event callback
tcccSDK.addTcccListener(self)
```

```
// Import the OC header file
#import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"

- (void)onNewSession:(TXSessionInfo *)info {
    // New session event, including inbound and outbound calls
}
- (void)onEnded:(TXEndedReason)reason reasonMessage:(NSString *_Nonnull)reasonMessa
    // Call end event
}
- (void)onAccepted:(NSString *_Nonnull)sessionId {
    // Remote party answered event
}
// Set the TCCC event callback
[self.tcccSDK addTcccListener:self];
```

```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;

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

};
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

Swift

Objective-C

C++

```
import TCCCSDK

func onConnectionLost(_ serverType: TXServerType) {
    // The connection between the SDK and the cloud has been disconnected
}
func onConnectionRecovery(_ serverType: TXServerType) {
    // The SDK's connection with the cloud has been restored
}
func onTry(toReconnect serverType: TXServerType) {
    // The SDK is trying to reconnect to the cloud
}

// Set the TCCC event callback
tcccSDK.addTcccListener(self)
```

```
// Import the OC header file
#import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"

- (void)onConnectionLost:(TXServerType)serverType {
    // The connection between the SDK and the cloud has been disconnected
}
- (void)onTryToReconnect:(TXServerType)serverType {
    // The SDK is trying to reconnect to the cloud
}
- (void)onConnectionRecovery:(TXServerType)serverType {
    // The SDK's connection with the cloud has been restored
}
// Set the TCCC event callback
[self.tcccSDK addTcccListener:self];
```

```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;

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

uni-app

Last updated：2025-01-09 15:41:51

API Overview

Creating Instances and Event Callbacks

API	Description
sharedInstance	Creates a TCCC Workstation instance (singleton).
destroyInstance	Destroys a TCCC Workstation instance (singleton). It is recommended to uninstall the TCCCWorkstation instance when it is not in use.
on	Sets a TCCC Workstation event callback.
off	Cancels a TCCC Workstation 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
    }
});
// Logout
tcccSDK.logout((code,message) => {
    if (code == TcccErrorCode.ERR_NONE) {
    // Logout succeeded
  } else {
    // Logout 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	Mute
unmute	Unmute
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	SDK and the cloud has been disconnected.
onTryToReconnect	SDK is trying to reconnect to the cloud.
onConnectionRecovery	SDK and the cloud connection have 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.

Outbound Integration Guide
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>
```

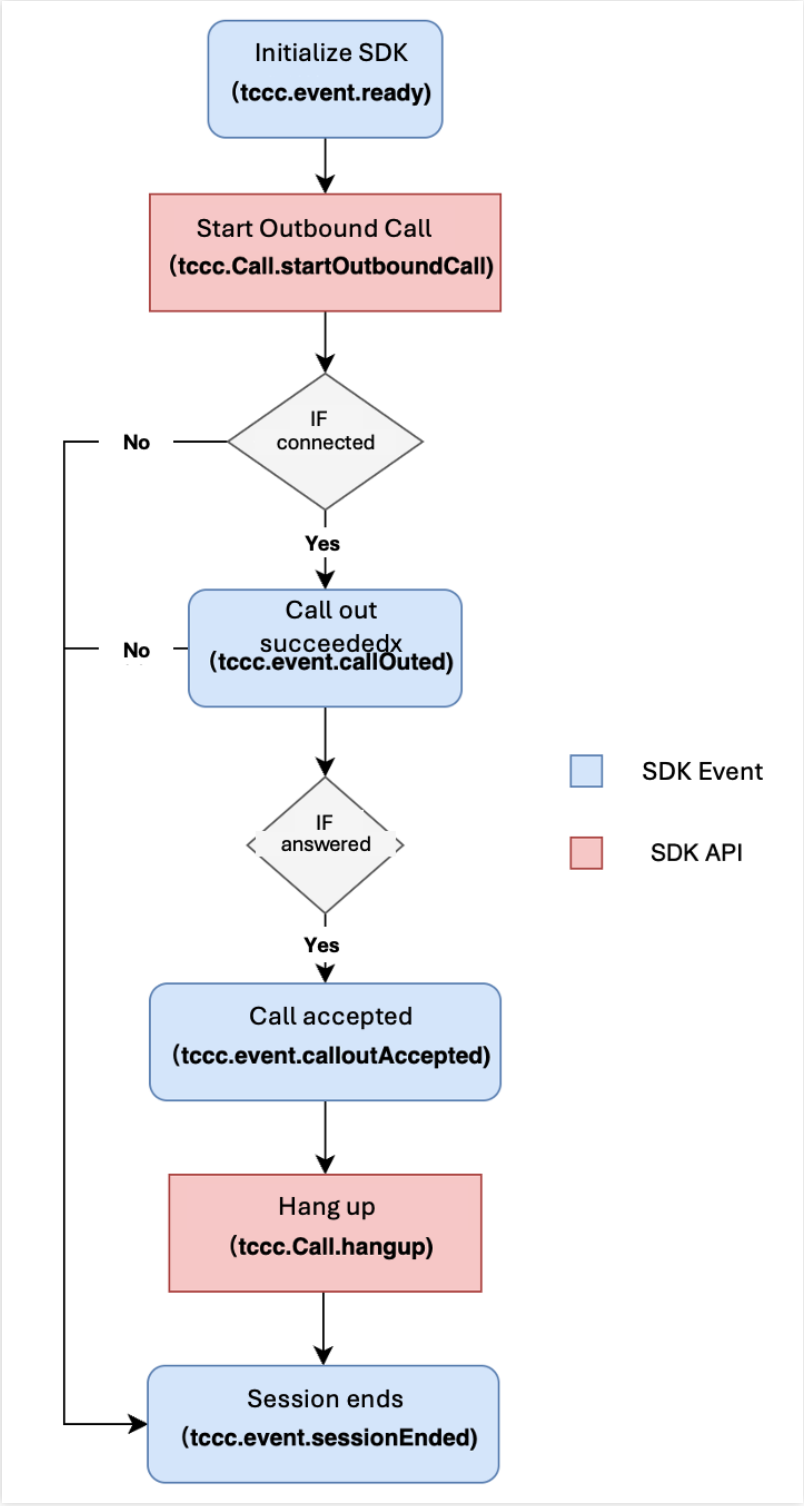
```
import { useState } from 'react';
```

```
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').addEventListsner('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


Initialize SDK
(tccc.event.ready)
Start Outbound Call
(tccc.Call.startOutboundCall)
IF
connected
No
Yes
Call out
succeededx
(tccc.event.callOuted)
No
IF
answered
Yes
Call accepted
(tccc.event.calloutAccepted)
Hang up
(tccc.Call.hangup)
Session ends
(tccc.event.sessionEnded)
SDK Event
SDK API

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

```
TCCCTypeDef.TCCCStartCallParams callParams =new TCCCTypeDef.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 TCCCCommonCallback : public ITXCallback {
private:
    NSString* mFunName;
public:
    TCCCCommonCallback(NSString* funName) {
        mFunName = funName;
    }
    ~TCCCCommonCallback() override {

    }
    void OnSuccess() override {
      // Succeeded
    }

    void OnError(TCCCError error_code, const char *error_message) override {
        std::string copyErrMsg = makeString(error_message);
      // Failed
    }
};
TCCCCommonCallback* startCallCallbackImpl = nullptr;
if (nullptr == startCallCallbackImpl) {
  startCallCallbackImpl = new TCCCCommonCallback(@"startCall");
}
TCCCStartCallParams 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();
```

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
```

```
    }
});
```

Inbound Integration Guide
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

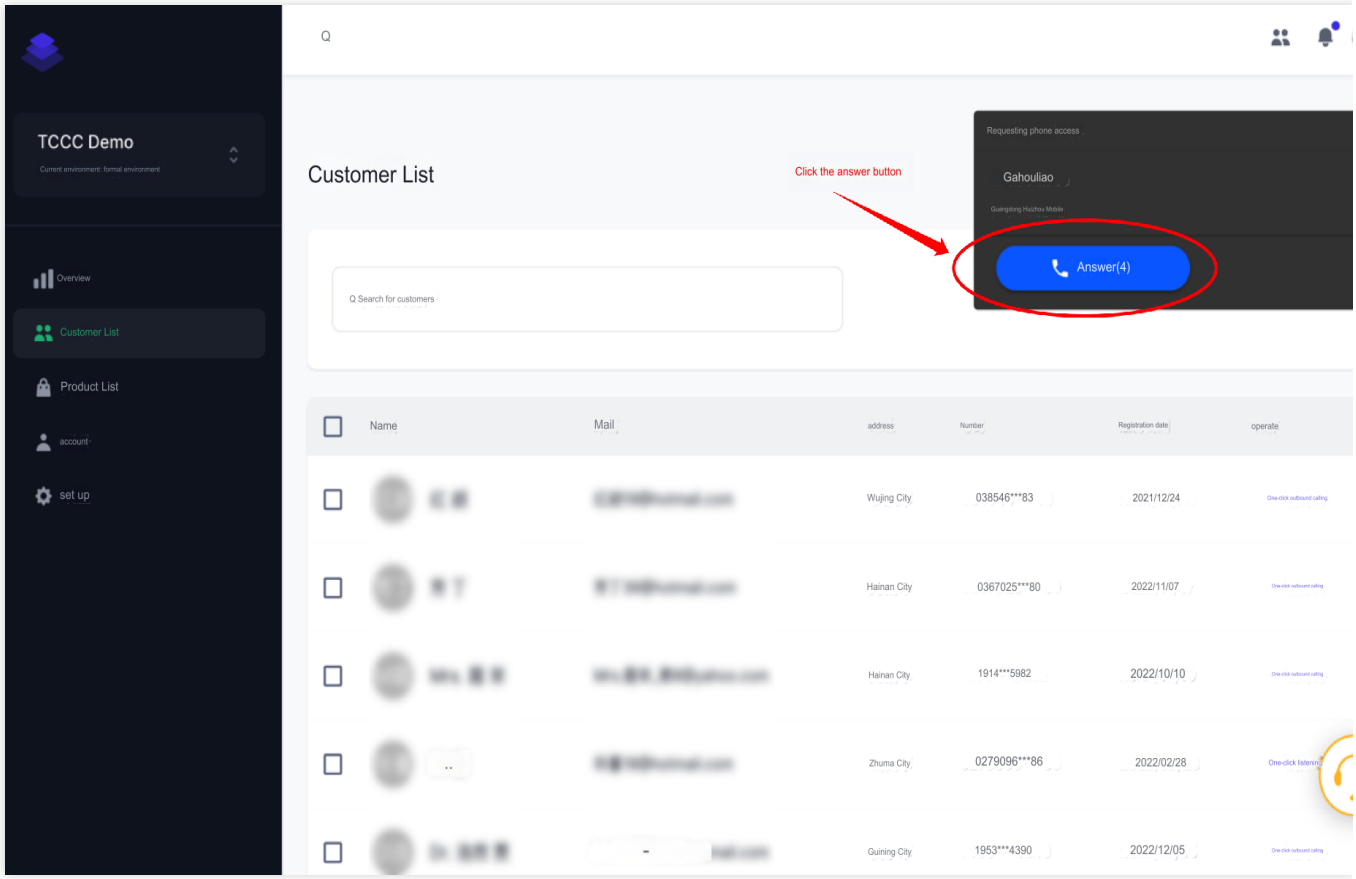
1. Bind the inbound call event `tccc.events.callIn` through `tccc.on` to monitor the inbound call and obtain sessionId;
2. Use `tccc.Call.accept()` to answer actively.

Sample code:

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

// 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 pub
  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 ca
```

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
})
```

FAQs
Web SDK FAQs

Last updated：2025-01-09 15:51:30

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?

No.

Can call toolbar button in the SDK be hidden?

Yes

What is UserID when 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?

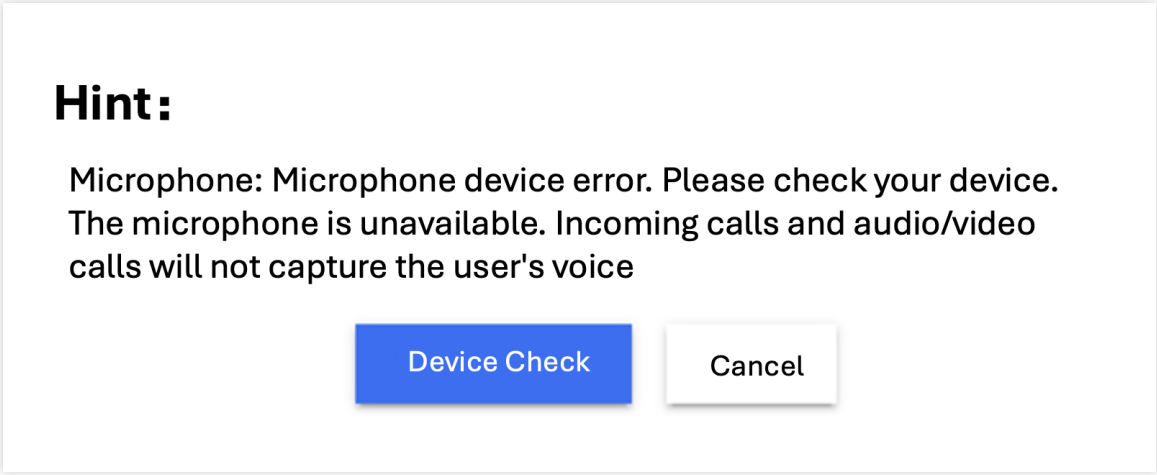
It is not supported on the interface. You can specify the display number when calling the outbound API of the SDK.

Token renewed? What if it expires?

After SDK initialization, Token no 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.



Why inbound call not ringing?

If the SDK page is minimized or switched during an inbound, browser's restrictions may cause the ringtone to be muted. It is recommended to enable browser notifications or use the `SDK callIn` event to implement a strong alert on the business side.

Why outbound call failed?

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 call disconnected unexpectedly?

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

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.

Uni-app FAQs

Last updated：2025-01-09 15:51:30

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.

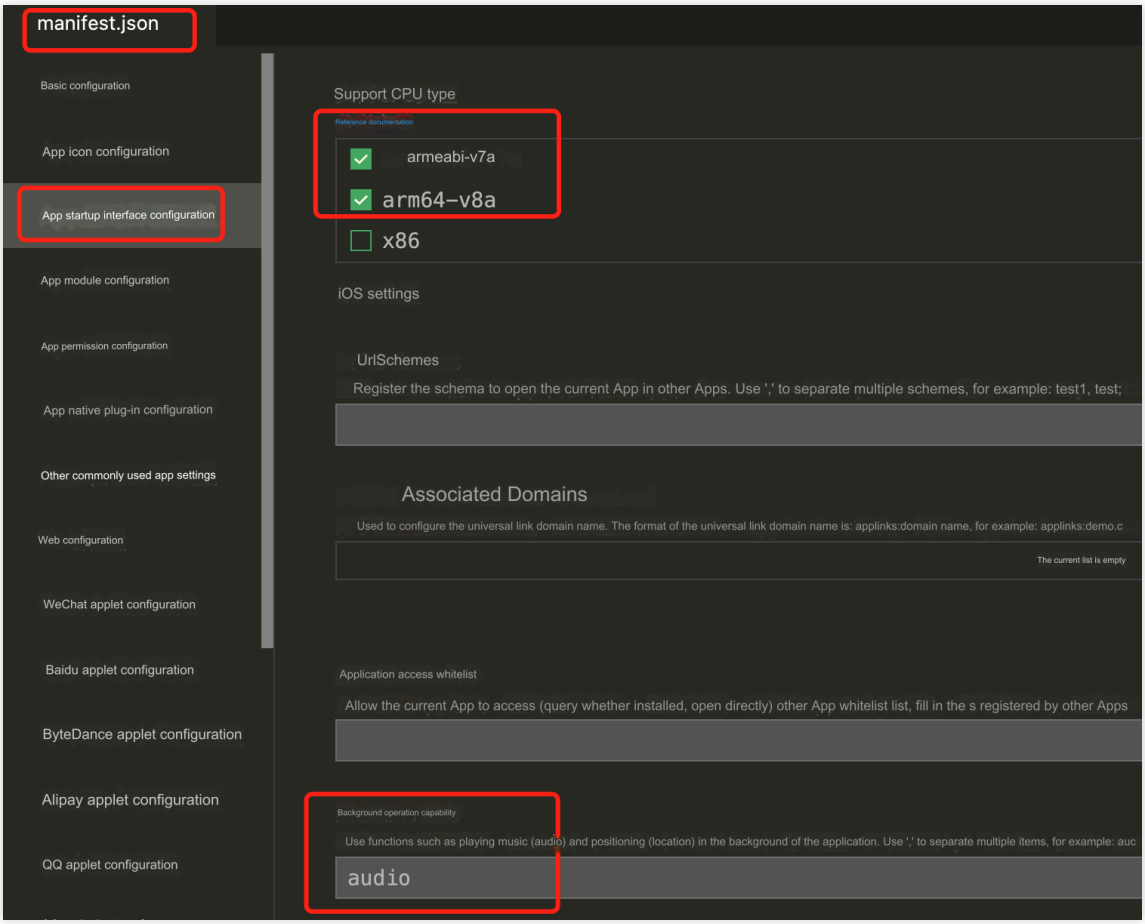
Does the Cloud Contact Center SDK support the CPU type armv7 on iOS?

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.

Link WhatsApp to Desk

Last updated：2025-04-11 17:51:38

Desk supports WhatsApp access. After access, it will host the messages sent by managed users to WhatsApp.

Integration Steps

1. Click and enter WhatsApp Group, and submit your request.

Caution:

Since the feature is still in beta test, you can only use it after applying for enabling it.

2. After enabling and completing configuration in Tencent Cloud SMS console, access management panel, click **Client Configuration > WhatsApp** in the left sidebar, and click **Link WhatsApp**.

3. In the pop-up window, fill in the required information correctly:

PhoneNumber. Please enter the mobile phone number that has been bound to the WABA ID in Tencent Cloud SMS console > WhatsApp Messages.

SMS SDKAppID. Go to Tencent Cloud SMS console > Application management > Application list to obtain the SMS SDKAppID.

SecretId. See Obtain SecretId and SecretKey.

SecretKey. See Obtain SecretId and SecretKey.

4. Refresh the page after successful authorization. Display the already bound WhatsApp PhoneNumber, indicating that the linking has been successful.

Get SecretId and SecretKey

Considering business security, it is recommended that you separately create SecretId and SecretKey for the WhatsApp channel. The specific acquisition method is as follows:

Create Policy

1. Go to Cloud Access Management, and in the left sidebar, click **Policies > Create Custom Policy**.

2. In the pop-up window for creating a policy, select create by policy generator.

3. In the Visual Policy Generator, select **SMS** service, check **all operations**, and select **all resources**.

Create a User and Bind a Policy

1. Access Management: In the left sidebar, click **Users** > **List of Users** > **Create User**.

2. Bind a policy for the newly created user.

Generate SecretId and SecretKey

1. In the left sidebar of Cloud Access Management, click **Users** > **List of Users**. Click the name of the newly created user to enter the user details page. Click **API key**.

2. Click **Create Key** to obtain SecretId and SecretKey.

Bind WhatsApp Online Session

Last updated：2025-05-15 17:07:01

Desk supports WhatsApp access. After access, it will host the messages sent by managed users to WhatsApp.
Note:
When your business has multiple consultation entry points, it is recommended that you choose this method to unify the user inquiries from multiple channels (for example: in-APP customer service, WhatsApp, etc.) of your business to our workbench. The final achieved effect is that employees can reply to inquiries from multiple channels simultaneously in one workbench, enhancing the reply efficiency.

Integration Steps

1. Click and enter WhatsApp Group, then submit your request.
2. After activation, please complete the configuration on the Tencent Cloud SMS console according to the access process.
3. After completing the configuration in Tencent Cloud SMS console, visit management panel, click **Client Configuration > WhatsApp** in the left sidebar, and click **Link WhatsApp**.

4. In the pop-up window, fill in the required information correctly:
PhoneNumber. Please enter the mobile phone number that has been bound to the WABA ID in Tencent Cloud SMS console > WhatsApp Messages.
SMS SDKAppID. Go to Tencent Cloud SMS console > Application management > Application list to obtain the SMS SDKAppID.
SecretId. See Get SecretId and SecretKey.
SecretKey, refer to Get SecretId and SecretKey.

5. Refresh the page after successful authorization. Display the already bound WhatsApp PhoneNumber, indicating that the linking has been successful.

Get SecretId and SecretKey

For business security, it is recommended that your business separately create SecretId and SecretKey for the WhatsApp channel. The specific acquisition method is as follows:

Creating Policies

1. Go to Cloud Access Management, and in the left sidebar, click **Policies** > **Create Custom Policy**.

2. In the pop-up window for creating a policy, select create by policy generator.

3. In the Visual Strategy Generator, select the SMS service, check all operations, and select all resources.

Create a User and Bind a Policy

1. Access Management: In the left sidebar, click **Users** > **List of Users** > **Create User**.

2. Bind a policy to the newly created user.

Generate SecretId and SecretKey

1. In the left sidebar of Cloud Access Management, click **Users** > **List of Users**. Click the name of the newly created user to enter the user details page. Click **API key**.

2. Click **Create Key** to obtain the SecretId and SecretKey.

Data Push
Preliminary Explanation (Data Push)

Last updated：2025-01-17 18:16:27

Data Push Configuration Steps

Tencent Cloud Contact Center can push service records and call recordings to a specified URL. Follow these steps to enable this feature:. For push format, see Data Push - Phone CDR Data Push.

1. Log into Tencent Cloud console, choose your Cloud Contact Center application > click Data Push.

2. In the data push settings, click **Modify**, and turn on the **Data Push** switch:

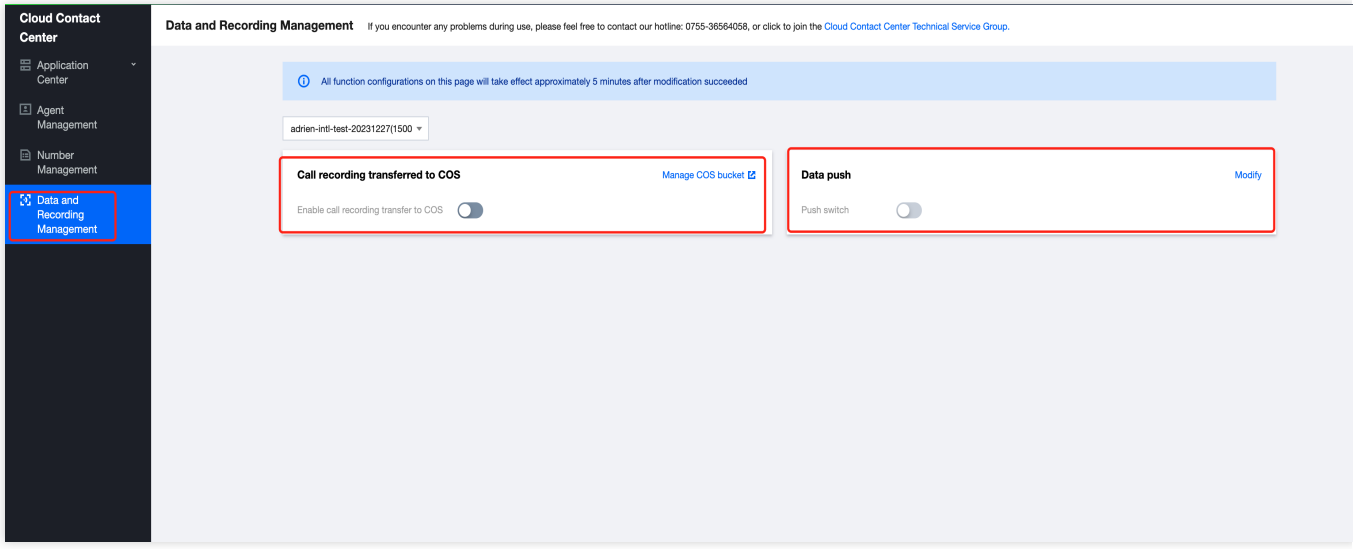
Push Address: Refer to About Third-Party Provided URL in the Preliminary Explanation of Data Push.

Authentication Approach: Refer to About Authentication in the Preliminary Explanation of Data Push.

CDR Data: For details on the push protocol, see Voice Call Data Push.

Call recording data: For details on the push protocol, consult Voice Call Data Push.

Voice mailbox data push: For details on the push protocol, refer to Voice Message Data Push.



Third-Party Provided URL

A third party provides a publicly accessible HTTP/HTTPS (HTTPS recommended) POST interface. Cloud Contact Center will push data to this interface and distinguish different data types through the URL parameter action.

Authentication

Currently supported authentication methods include:
1. Authentication-Free: No extra authentication.
2. basicAuth: Corresponding to the **Account Password** Setting in the Settings menu, where the username is the username and the password is the password.
3. OAuth2.0 client credentials: Corresponding to "OAuth2.0" in the Settings menu. The parameters that need to be configured include the URL of the token, ClientID, and ClientSecret.

Return Value Format

The return format should be json type, following the format specified in the API documentation. If successful, the ErrCode field needs to be set to 0. Otherwise, Cloud Contact Center will try to repush the data, and the maximum retry times is 3.

Duplicate and Out-of-Order Data Handling

The X-TCCC-PUSH-UUID header is used to uniquely identify a data push. If the same X-TCCC-PUSH-UUID is received due to retries from the sender, the receiver should handle deduplication accordingly.

Push Failure Handling

Pushes may fail or be delayed due to network issues or receiver instability. For scenarios requiring data consistency and timeliness, it is recommended to regularly call Obtaining telephone service records and recordings API to retrieve and reconcile call service records and recordings.

Data Push: Voice Call Records

Last updated：2025-06-24 18:10:41

CDR records data based on the entire session. A complete customer inbound or outbound call corresponds to a record. The root-level data indicators of CDR represent the global information of the customer-dimensional session. The specific detail tracks in the session service are described through the ServeParticipants object array (such as: phone transfer information). Each piece of ServeParticipants data represents a service track.
Oubound call type data QueuedSkillGroupID field selection strategy:
Customer service belongs to only one phone skill group, then hit.
Customer service belongs to multiple phone skill groups, preferentially select the skill group bound to the outbound number (take the first skill group if multiple).
If failing to satisfy 1 and 2, take the first phone skill group of customer service.
URL: `https://{custom_url}?action=cdr&version=1`
METHOD: `POST`
Content-Type: `application/json;charset=utf8`
REQUEST:

Parameter	Type	Description
SdkAppId	Long	The unique Cloud Contact Center application ID, you could find this info on Console.
SessionId	String	Conversation ID
Caller	String	caller
Callee	String	callee
Direction	Integer	Call Direction: inbound call Outgoing calls
CallType	Integer	Call Type: 1. Voice outbound call 2. Voice Inbound call 3. Audio Inbound 5 Predictive Dialing Call 6. Internal Call between Staff Example value: 1
Duration	Integer	Session overall service time, unit: seconds; EndedTimestamp-AcceptTimestamp
SeatUser	Object	Customer service information, format as follows (if transferred, the last customer service info)
CallerLocation	String	Caller phone number location

IVRDuration	Integer	IVR stage duration in seconds, QueuedTimestamp - StartTimestamp
RingTimestamp	Integer	When the session direction is inbound, it means the agent side ring timestamp (UNIX second-level timestamp). When the session direction is outbound, it means the user-side ring timestamp (UNIX second-level timestamp).
AcceptTimestamp	Integer	Timestamp of the start of the agent answering the call when the session direction is inbound (UNIX second-level timestamp) Timestamp of the start of the user answering the call when the session direction is outbound (UNIX second-level timestamp)
EndedTimestamp	Integer	Session end timestamp (UNIX second-level timestamp)
StartTimestamp	Integer	Entire session start timestamp (UNIX second-level timestamp)
IVRKeyPressed	String array	IVR key information (e.g. ["1","2","3"])
IVRKeyPressedEx	Object array	IVR key information (e.g. [{"Key":"1","Label":"super satisfied"}])
HungUpSide	String	Hang Up Party (user - client hang up or seat - agent hang up)
ServeParticipants	Object array	List of service participants. The format is shown in the table below.
EndStatusString	String	The session end status. For details, see EndStatusString
QueuedTimestamp	Integer	When the session direction is inbound, the time when the user enters the queue
PostIVRKeyPressed	Object array	post IVR key information (e.g. [{"Key":"1","Label":"super satisfied"}])
QueuedSkillGroupName	String	When the session direction is inbound, the skill group name where the user enters the queue
QueuedSkillGroupId	Integer	Session enters Queue Skill Group ID
RecordId	String	Recording ID for user-side recording
UserRemark	String	user remark
Uui	String	Accompanied data (data entered by the customer via the telephone outbound interface)
TelLocation	Json object	Number location information, format as below

SeatUser data format:

Parameter	Type	Description
Mail	String	agent email
Name	String	Agent Name
Nick	String	agent nickname
Phone	String	agent telephone number
UserId	String	User ID
StaffNumber	String	Agent ID
SkillGroupNameList	String array	Affiliated skill group list of the agent

ServeParticipants data format:

Parameter	Type	Description
Mail	String	agent email
Phone	String	agent phone
RingTimestamp	Long	Ring timestamp, Unix second-level timestamp
AcceptTimestamp	Long	Answer timestamp, Unix second-level timestamp
EndedTimestamp	Long	end timestamp, Unix second-level timestamp
RecordId	String	Recording ID
Type	String	participant type staffSeat outboundSeat staffPhoneSeat miniProgramSeat
TransferFrom	String	transfer source agent information
TransferFromType	String	transfer source agent type
TransferTo	String	transfer destination agent information
TransferToType	String	transfer destination participant type, consistent with Type value
SkillGroupId	Integer	skill group ID

EndStatusString	String	Session participant end status. For details, see EndStatusString
Sequence	Integer	participant sequence number, starting from 0
StartTimestamp	Long	start timestamp, Unix second-level timestamp
SkillGroupName	String	Group name
SkillGroupPriority	Integer	Skill group assignment priority

TelLocation data format:

Parameter	Type	Description
TelNumber	String	number
Country	String	Country
Province	String	Province
City	String	City
Operator	String	Carrier

RESPONSE:

Parameter	Type	Description
ErrMsg	String	Error description
ErrCode	Integer	Error code

Sample data:

```
{
  "SessionId": "99a1c8f8-eb3d-4xxx-8401-5f6aa8761232",
  "Caller": "0086400xxx6666",
  "Callee": "0086184xxxx7605",
  "Direction": 1,
  "Duration": 0,
  "SeatUser": {
    "Mail": "zhangsan@tencent.con",
    "Name": "Zhang San"
    "Nick": "Youyou",
    "Phone": "",
    "UserId": "zhangsan@tencent.com",
    "StaffNumber": "8546",
```

```
      "SkillGroupNameList": [
        "consultant outbound call"
      ]
    },
    "CallerLocation": "",
    "IVRDuration": 0,
    "RingTimestamp": 1677140072,
    "AcceptTimestamp": 0,
    "EndedTimestamp": 1677140081,
    "IVRKeyPressed": null,
    "IVRKeyPressedEx": null,
    "HungUpSide": "seat",
    "ServeParticipants": [
      {
        "Mail": "zhangsan@tencent.com",
        "Phone": "",
        "RingTimestamp": 1677140068,
        "AcceptTimestamp": 1677140069,
        "EndedTimestamp": 1677140081,
        "RecordId": "dbe87035-019c-4xxx-bf4f-c29701ad315d",
        "Type": "miniProgramSeat",
        "TransferFrom": "",
        "TransferFromType": "",
        "TransferTo": "",
        "TransferToType": "",
        "SkillGroupId": 2734,
        "EndStatusString": "ok",
        "Sequence": 0,
        "StartTimestamp": 1677140068,
        "SkillGroupName": "consultant outbound call",
        "SkillGroupPriority": 0
      }
    ],
    "EndStatusString": "numberNotExist",
    "StartTimestamp": 1677140068,
    "QueuedTimestamp": 0,
    "PostIVRKeyPressed": null,
    "QueuedSkillGroupId": 2734,
    "QueuedSkillGroupName": "consultant outbound call",
    "SdkAppId": 1400482256,
    "RecordId": "f65472d9-400a-4xxx-a51f-a49a55dab99a",
    "UserRemark": "*******7605",
    "Uui": "abc",
    "TelLocation": {
      "TelNumber": "008618486147605",
      "Country": "China",
      "Province": "Guizhou",
```

```
      "City": "Anshun",
      "Operator": "mobile"
    }
  }
```

Data Push: Voice Call Recording

Last updated：2025-01-15 11:08:11

Duo Track Recordings are created for each participant in a session. A normal call has two recordings (caller and callee) with the same `SessionId`, distinguished by `EndpointUser`. If a call is transferred, a third recording is added for the transfer agent. Use `SessionId`, `EndpointUser`, and `RecordId` to find each recording URL.

URL: `https://{custom_url}?action=record&version=1`

METHOD: `POST`

Content-Type: `application/json;charset=utf8`

REQUEST:

Parameter	Type	Description
SdkAppId	Numerical value (long integer)	The unique Cloud Contact Center application ID, you could find this info on Console.
RecordId	String	Recording ID
SessionId	String	Session ID
Timestamp	Numerical value (long integer)	Recording generation timestamp
EndpointUser	String	Recording object (the user's mobile phone number or the agent's email)
RecordURL	String	Recording URL (free storage for 3 months by default)
CustomRecordURL	String	Recording can be saved to your COS URL (This field is available only when the recording save function is enabled).

RESPONSE:

Parameter	Type	Description
ErrMsg	String	Error description
ErrCode	Numerical value	Error code

Data sample:

```
{
    "SdkAppId": 1400264214,
    "RecordId":"1608130650",
```

```
    "SessionId": "e97be0ab-1ef6-4ad2-a8c4-2b2bbfb18e55",
    "Timestamp": 1608130650,
    "EndpointUser": "lululing@tencent.com",
    "RecordURL": "http://recorder-10018504.cos.ap-shanghai.myqcloud.com/def/month12
}
```

Data Push: Voicemail Recordings

Last updated：2025-01-15 11:08:11

Voicemail recordings will include caller recordings generated by IVR voicemail module, and you can use the SessionId to find the URL of the voicemail recording.

URL: `https://{custom_url}?action=voicemail`

METHOD: `POST`

Content-Type: `application/json;charset=utf8`

REQUEST:

Parameter	Type	Description
SdkAppId	Numerical value (long integer)	The unique Cloud Contact Center application ID, you could find this info on Console.
SessionId	String	Session ID
Timestamp	Numerical value (long integer)	Recording generation timestamp
RecordURL	String	Recording URL
EndpointUser	String	Recording object (the user's mobile phone number)

RESPONSE:

Parameter	Type	Description
ErrMsg	String	Error description
ErrCode	Numerical value	Error code

Data sample:

```
{
    "SdkAppId": 1400264214,
    "SessionId": "e97be0ab-1ef6-4ad2-a8c4-2b2bbfb18e55",
    "Timestamp": 1608130650,
    "RecordURL": "http://recorder-10018504.cos.ap-shanghai.myqcloud.com/def/month12
    "EndpointUser": 13123456789
}
```