Tencent Cloud Super App as a Service

Guidelines for Code Integration

Product Documentation

Contents

Guidelines for Code Integration
Get Demo and SDK

Last updated：2025-03-20 18:08:14

Android

Online integration method Demo acquisition

Offline integration method Demo and SDK acquisition

iOS

Note：

Follow Step One: Getting the Application SDK Configuration., replace it and run it. The project BundleId must be consistent with the BundleId set by the operating platform.

Get the address through CocoaPods integrated Demo

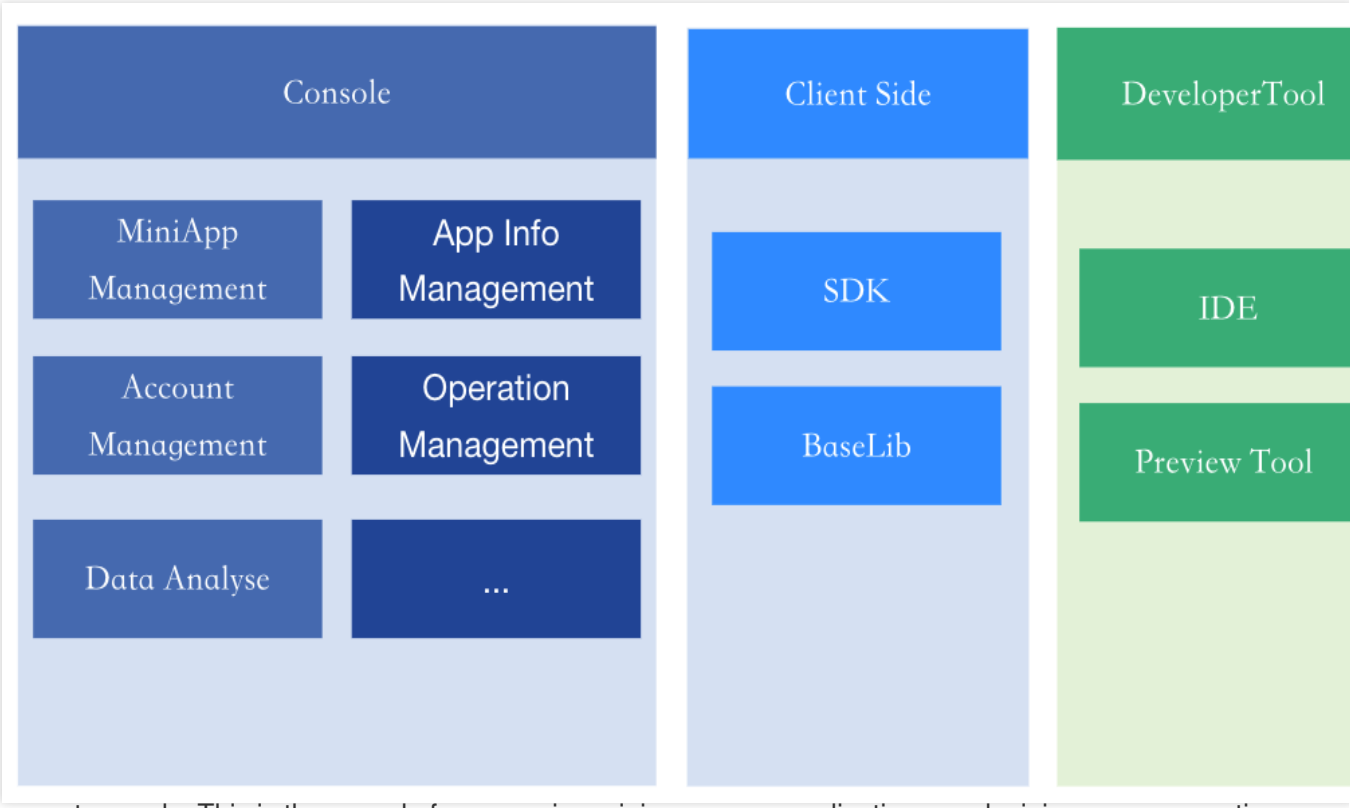
Manually integrate Demo and sdk to obtain the address

Android
Android SDK Description
SDK Introduction

Last updated：2025-03-20 18:08:15

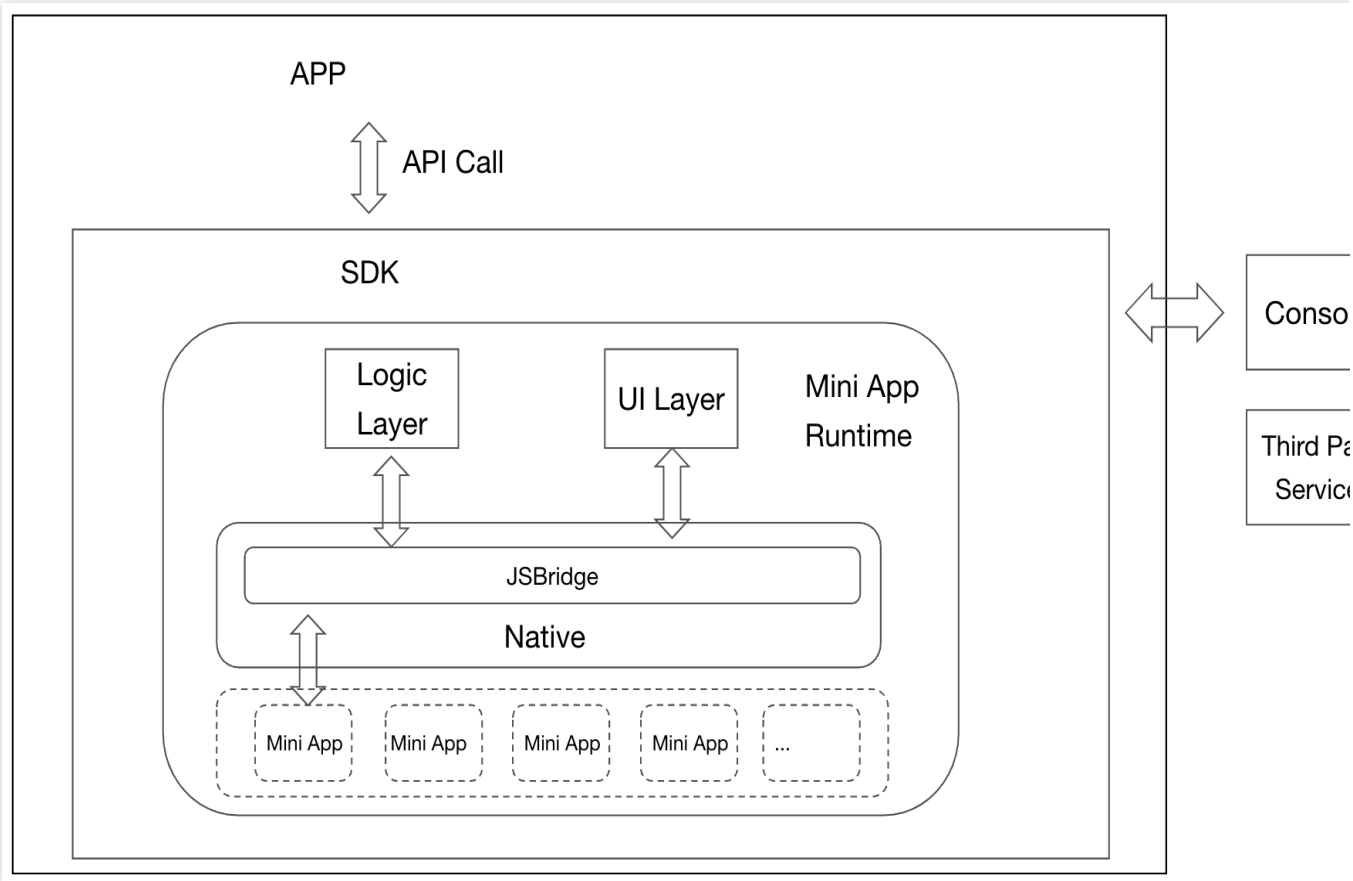
Product Introduction

Our product consists of three parts: management console, client SDK and mini program development tool (IDE).



Management console: This is the console for managing mini programs, applications, and mini program operations.
Client: This refers to the mobile client application integrated with the mini program SDK. The SDK provides a runtime environment for mini programs within the client application.
Developer tools: Developers use the mini program IDE for developing, debugging, and submitting versions of mini programs. The mini program preview assistant is used for previewing and validating mini programs on mobile clients.

How the mini program SDK Works



The mini program SDK provides a runtime environment for mini programs within client apps. It communicates with the product backend to fetch mini program information and loads and runs the mini programs in the provided runtime environment.

Mini program SDK description

To keep the SDK lightweight and modular, it is divided into two categories: core SDK (mini_core) and extension SDKs.

Core SDK: Provides the minimal environment required to load and run mini programs. Users must integrate the core SDK to utilize basic mini program capabilities.

Extension SDKs: Provide additional features that users can integrate based on their needs.

Mini program SDKs:

SDK name	Description	Usage notes	Package size
Core SDK (mini_core)	Provides the minimal runtime environment for loading mini programs.	-	~4.7MB
Same-layer rendering	Enables same-layer	Requires public version	~55 KB (excluding x5

extension SDK - Public version (mini_extra_public_x5)	rendering for mini program components.	kernel. For details, see SDK extension components.	kernel)
Same-layer rendering extension SDK - Dynamic version (mini_extra_dynamic_x5)	Enables same-layer rendering for mini program components.	Requires dynamic version kernel. For details, see SDK extension components.	~56.7 KB (excluding x5 kernel)
Same-layer rendering extension SDK - Static version (mini_extra_static_x5_new)	Enables same-layer rendering for mini program components.	Requires static version kernel. For details, see SDK extension components.	~54.4 KB (excluding x5 kernel)
QR code extension SDK (mini_extra_qrcode)	Provides QR code scanning capabilities.	For details, see SDK extension components	~ 35 KB
Map extension SDK - Tencent Maps version (mini_extra_map)	Supports Tencent Maps for location and map services.	Requires Tencent Maps SDK and dependencies. For details, see SDK extension components	About 156.4 KB, the volume does not include Tencent Map SDK
Map extension SDK - Huawei Maps version (mini_extra_huawei_map)	Supports Huawei Maps for location and map services.	Requires Huawei Maps SDK and dependencies. For details, see SDK extension components	~236.8KB (excluding Huawei Maps SDK)
Map extension SDK - Google Maps Version (mini_extra_google_map)	Supports Google Maps for location and map services.	Requires Google Maps SDK and dependencies. For details, see SDK extension components	~153.2KB (excluding Google Maps SDK)
Live streaming extension SDK (mini_extra_trtc_live)	Provides live streaming capabilities for mini programs.	Requires Tencent TRTC dependencies. For details, see SDK extension components	~91.4KB (excluding Tencent TRTC SDK)
LBS extension SDK (mini_extra_lbs)	Provides location-based services for mini programs.	For details, see SDK extension components	~ 73.8 KB
Bluetooth extension SDK (mini_extra_bluetooth)	Provides Bluetooth API access for mini programs.	For details, see SDK extension components	~ 115 KB
NFC extension SDK (mini_extra_nfc)	Provides NFC API access for mini	For details, see SDK extension components	~ 65 KB

	programs.		
Biometric authentication extension SDK (mini_extra_soter)	Provides biometric authentication API access for mini programs.	For details, see SDK extension components	~ 71 KB
Clipboard extension SDK (mini_extra_clipboard)	Provides clipboard access for mini programs.	For details, see SDK extension components	~7KB
Contacts extension SDK (mini_extra_contact)	Provides contacts API access for mini programs.	For details, see SDK extension components	~ 49 KB
Document engine extension SDK (mini_extra_doc)	Provides document preview capabilities for mini programs.	Requires Tencent Document SDK. For details, see SDK extension components	~40KB
Media extension SDK (mini_extra_media_support)	Provides media selection API access for mini programs.	For details, see SDK extension components	~ 568.1 KB

SDK Integration

Last updated：2025-06-06 18:27:18

This doc will guide client developers in integrating a minimal functionality Mini Program SDK to enable the opening of mini programs.

Integration process

1. Configure the online repository address

The method for configuring the repository address depends on the version of the Gradle plugin used in your current project:
For Gradle version earlier than 7.0, see Method 1.
For Gradle version 7.1 and later, see Method 2.
Method 1: For Gradle version earlier than 7.0
Add the following configurations to the project-level build.gradle file:

```
buildscript {
    dependencies {
        classpath 'org.jetbrains.kotlin:kotlin-gradle-plugin:1.4.32'
    }
}

allprojects {
    repositories {
        maven {
            url 'https://maven-dev.tcmppcloud.com/fHKFBbEjd/repository/maven-public
        }
        maven {
            url 'https://tcmpp-work-maven.pkg.coding.net/repository/tcmpp/android'
        }
    }
}
```

Method 2: For Gradle version 7.1 and later
Add the following configurations to the project-level settings.gradle file.

```
  pluginManagement {
    repositories {
        maven {
            url 'https://maven-dev.tcmppcloud.com/fHKFBbEjd/repository/maven-public
        }
```

```
        maven {
            url 'https://tcmpp-work-maven.pkg.coding.net/repository/tcmpp/android'
        }
    }
}
dependencyResolutionManagement {
    repositories {
        maven {
            url 'https://maven-dev.tcmppcloud.com/fHKFBbEjd/repository/maven-public
        }
        maven {
            url 'https://tcmpp-work-maven.pkg.coding.net/repository/tcmpp/android'
        }
    }
}
```

2. Kotlin plugin configuration

The configuration of the Kotlin plugin also depends on the version of the Gradle plugin used in your project.

Method 1: For Gradle version earlier than 7.0

```
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-kapt'
```

Method 2: For Gradle version 7.1 and later

```
  plugins {
    id "org.jetbrains.kotlin.android"
    id "org.jetbrains.kotlin.kapt"
}
```

3. Configure packagingOptions

Add the following configurations to the app-level build.gradle file.

```
android {
    defaultConfig {
        packagingOptions {
            pickFirst 'lib/arm64-v8a/libc++_shared.so'
            pickFirst 'lib/armeabi/libc++_shared.so'
            pickFirst 'lib/armeabi-v7a/libc++_shared.so'
            pickFirst 'lib/arm64-v8a/libmarsxlog.so'
            pickFirst 'lib/armeabi/libmarsxlog.so'
            pickFirst 'lib/armeabi-v7a/libmarsxlog.so'
            pickFirst 'lib/arm64-v8a/libv8jni.so'
        }
```

```
    }
}
```

4. Configure mini program SDK dependencies

```
dependencies {
    implementation 'com.google.android.material:material:1.3.0-alpha03'
    implementation 'androidx.core:core-ktx:1.6.0'

    //gosn
    implementation 'com.google.code.gson:gson:2.8.6'
    // ok-http
    implementation 'com.squareup.okhttp3:okhttp:3.12.13'

    // mini app start
    kapt 'com.tencent.tcmpp.android:mini_annotation_processor:${version}' // For ve
    implementation 'com.tencent.tcmpp.android:mini_core:${version}' // For version
    // mini app end
}
```

Note:

Replace {version} in the configuration with the corresponding version of the dependencies. For version information, please refer to the Android SDK Updates.

If your existing project uses annotationProcessor, you will need to change all instances of annotationProcessor to kapt.

Once the dependencies for the Mini Program SDK are added, you can begin integrating the Mini Program SDK as described in the following sections.

Preparation before integration

1. Obtain the configuration file

The initialization of the mini program SDK relies on `a configuration file` from the `mini program console.` Before integrating the Mini Program SDK, you need to retrieve this configuration file.

The process for obtaining the configuration file is as follows:

Log in to the mini program console

Create an app

Fill in the app information

Download the configuration file

Note:

The default name of the downloaded configuration file is tcsas-android-configurations.json.

2. Add the configuration file to the project

After obtaining the configuration file, you need to copy the `configuration file` to the assets directory of your project.

Note:

Ensure the app package name matches the packageName in the configuration file. Otherwise, the mini program SDK cannot pass the package name verification when running, which will cause a SDK initialization failure.
If there is a mismatch, you cannot directly modify the package name in the configuration file. Instead, you should correct it using one of the following methods:
Change the app package name in your project to match the package name in the configuration file.
Modify the app package name in the console to match the package name in your project and then re-download the configuration file.
The packageName field in the configuration file must be consistent with the package name of the current project.

3. Add mini program SDK initialization configuration in the source code

Extend and implement the MiniConfigProxy class.
Add the following annotations for the implementation class of MiniConfigProxy:

```
   @ProxyService(proxy = MiniConfigProxy.class)
```

Example:

```
@ProxyService(proxy = MiniConfigProxy.class)
public class MiniConfigProxyImpl extends MiniConfigProxy {
    /**
     * Returns the Application instance of the superapp
     * @return
     */
    @Override
    public Application getApp() {
        // Must use the Application instance of the superapp
        return "app Application";
    }

      /**
     * Create initialization configuration information
```

```
     * @return
     */
    @Override
    public MiniInitConfig buildConfig() {
      MiniInitConfig.Builder builder = new MiniInitConfig.Builder();
      MiniInitConfig config = builder
                .configAssetName("tcsas-android-configurations.json") // Configurat
                .autoRequestPermission(true) // Whether to automatically request sy
                .debug(true) // Log switch, default is off
                .build();
      return config;
    }
}
```

Note:

The configAssetName in MiniConfig is used to specify the path and name of the configuration file in the source code project.

At this point, the integration of the Mini Program SDK is complete. The next step is to use the APIs provided by the Mini Program SDK to open and preview mini programs.

Open a mini program

To open a mini program, use the following code:

Note:

The appid is the mini program appid, which needs to be obtained from the mini program developer.

```
TmfMiniSDK.startMiniApp(activity, appId, miniStartOptions);
```

SDK Integration FAQs

Last updated：2025-06-06 18:27:18

When integrating the mini program SDK, you may encounter the following AAPT issue during project compilation:

```
AAPT: error: attribute android:requestLegacyExternalStorage not found.
```

Solution:

Add the following configuration under the `<application> tag` in `AndroidMainifest.xml` :

```
<application
    android:theme="@style/AppTheme"
    tools:replace="android:icon"
      tools:remove="android:requestLegacyExternalStorage">
 /application>
```

If you see the following "Duplicate class android.support.v4" error during compilation:

```
Duplicate class android.support.v4.app.INotificationSideChannel found in modules co
```

Solution:

Add the following lines to your `gradle.properties` file:

```
android.useAndroidX=true
android.enableJetifier=true
```

If you encounter a version mismatch error like this during compilation:

```
Execution failed for task ':app:kaptGenerateStubsDebugKotlin'.
> 'compileDebugJavaWithJavac' task (current target is 1.8) and 'kaptGenerateStubsDe
  Consider using JVM toolchain: https://kotl.in/gradle/jvm/toolchain
```

Solution:

Modify the JDK version in the compileOptions section of your build.gradle file to match:

```
android {

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_17 // Match with kapt version
        targetCompatibility JavaVersion.VERSION_17 // Match with kapt version
    }
```

```
}
```

If you see the following error related to ProxyService:

Solution:

Check if you have the following configuration in your project and remove it if it exists:

```
kapt.include.compile.classpath=false
```

Minimum Android version supported by mini program SDK

The minimum SDK version (minSdkVersion) supported is 21, which corresponds to Android 5.0.

SDK initialization check

From version 1.5.1.1 of the mini_core, you can check if the SDK initialized successfully by filtering logs with MINI_SHARK. A successful initialization will log: 'shark init ok!'

If the initialization fails, an exception log will be displayed.

Supported devices for debugging

Supported: ARM architecture emulators and real devices. Not supported: x86 emulators.

Taro framework support

```
Uncaught DOMException: Failed to read the 'sessionStorage' property from 'Window':
    at <anonymous>:1207:96250
    at Array.forEach (<anonymous>)
    at Module.<anonymous> (<anonymous>:1207:96164)
    at Module.9 (<anonymous>:1207:113782)
    at l (<anonymous>:1203:566)
    at Module.204 (<anonymous>:1215:87806)
    at l (<anonymous>:1203:566)
    at t (<anonymous>:1203:435)
    at Array.r [as push] (<anonymous>:1203:298)
    at <anonymous>:1215:125
```

Add the following dependency if using the Taro framework:

```
implementation 'com.tencent.tcmpp.android:mini_extra_v8:${version}'
```

Troubleshooting mini program request errors

Errors during mini program operations like scanning, launching, or searching will be shown via Toast messages, including error codes and TraceId. These details are also logged for further investigation.
For 5-digit error codes, these are general client-side errors. For specific details, refer to Android Error Codes.
Common error codes include:
-11001: Shark network error. Check the network link and gateway between the client and server.
-11002 Server-side business logic error. Investigate the service based on the accessed API, and search the specific business error codes in the logs.
-11006 The mini program does not exist in the current configuration environment. Check if the mini program is released and the environment matches the app's configuration.
-12001 Shark network instance creation failed, usually due to missing, incorrect, or mismatched configuration files.
-12012 Mini program download failed. Ensure the download service is operational.
-12013 Mini program package parsing error. The package may be corrupted. Check for non-ASCII characters in resource file paths if using an older backend version.
For errors with different digit counts, they indicate specific business errors similar to -11002.
Use the TraceId to correlate and find the request details in backend logs for troubleshooting.

Remove unnecessary language resources from the SDK

The SDK supports multiple languages by default, including English, French, Arabic, and Indonesian. To remove other language resources, you can specify the supported languages in the build.gradle file of your app module by adding resConfigs. This will ensure that only the languages you need are included, while others are excluded.

```
android {
    defaultConfig {
        // Specify the languages you want to support; other languages will be exclu
        resConfigs "en", "fr"
    }
}
```

Android API
Mini Program Management APIs
Opening Mini Programs

Last updated：2025-05-26 15:22:54

The mini program management APIs provide capabilities for searching, opening, and closing mini programs.

Opening mini programs

When opening a mini program, the system will check whether there is a locally cached one. If not, it will automatically download the mini program from the remote server and then open it. If a cached version is available, the local mini program will be opened first while the system checks in the background for any new versions on the server.

Notes:

If a new version is available, it will be downloaded, and the next time you open the mini program, the updated version will be used.

1. Opening a mini program by mini program ID (appId)

To open the released version of a mini program :

Notes:

The appId field is the ID of the mini program, which can be obtained from the mini program developer or via the mini program search API.

The miniStartOptions are the startup parameters for the mini program. For details, see API Description.

```
TmfMiniSDK.startMiniApp(activity, appId, miniStartOptions);
```

To open the Preview or development version of a mini program:

```
TmfMiniSDK.startMiniApp(activiy, appId, MiniScene.LAUNCH_SCENE_MAIN_ENTRY, appVerTy
```

Notes:

The appVerType should match the version of the mini program. The value of the appVerType can be obtained from the MiniApp object instance returned by the API (getRecentList).

The value of appVerType for the Preview of the mini program should be MiniApp.TYPE_PREVIEW.

The value of appVerType for the development version of the mini program should be MiniApp.TYPE_DEVELOP.

2. Opening a mini program through QR code

The mini program SDK provides an API to open mini programs based on QR code scanning in the console. Before using the scanning capability provided by the mini program SDK, you need to integrate the scanning extension capability. For details about the integration, refer to the scanning capability documentation.
After the scanning capability is integrated, you can start the QR code scanning and open the mini program with one of the options:
Option 1: Process the scanning results by host app's Activity
Start QR code scanning:

```
TmfMiniSDK.scan(activity);
```

Get the result of the QR code scanning:
Notes:
The result of the QR code scanning is returned through the Activity's onActivityResult method. To identify the QR code, you need to rewrite the Activity's onActivityResult method and call `TmfMiniSDK.getScanResult` to get the QR code.

```
@Override protected void onActivityResult(int requestCode, int resultCode, @Nullabl
    // Get QR code scanning result  JSONObject scanResult = TmfMiniSDK.getScanResul
```

Open the mini program according to the QR code scanning result:
Notes:
The getScanResult method returns data in a JSON structure. The mini program's startup parameters are stored in the `'result'` field of this JSON structure. You need to extract the value of the `result` field and use it to open the mini program.

```
@Override protected void onActivityResult(int requestCode, int resultCode, @Nullabl
    //Get QR code scanning result  JSONObject scanResult = TmfMiniSDK.getScanResult
    if (scanResult != null) { // Extract `result` field String result = scanResult.
            //Use the value of the `result` field to open the mini program TmfMiniS
```

Option 2: Process the scanning results automatically by the agent Activity and start up the mini program directly
Scan the QR code and start up the mini program:

```
TmfMiniSDK.startMiniAppByScan(activity);
```

3. Configuring Scheme to open the mini program

Mini programs can be opened by third-party apps and scanning tools via URL redirection. Before you can use this capability, you need to configure the URL Scheme of the host app by adding the following string resource to the app:

```
<string name="mini_sdk_intent_filter_scheme">your scheme</string>
```

where `your scheme` should be replaced by the host's Scheme.

The host of the mini program URL is applet and the full format of the URL is: ${scheme}://applet?appid=${appId}&path=${encoded path}&param=${encoded param}. Parameters in the URL are as follows:

Parameter	Type	Required	Description
appId	String	Yes	Mini program ID
path	String	No	Mini program entry path, for which the URL encoding is required
param	String	No	The query passed to the mini program, for which the URL encoding is required

In the product console, a corresponding URL is created for the released version of each mini program and a QR code is generated based on the URL. When a URL is created, tcmpp + host app's appkey is used as the Scheme by default.

To ensure that a correct URL is generated, please replace it with the host app's URL Scheme. Click Modify to change the Scheme used to generate the URL:

Once configured, you can open the mini program by scanning the generated QR code or via URL redirection.

4. Opening a mini program by search

Call the mini program search API:

Notes:

The parameters for SearchOptions are:

keyword: Mini program keyword

firstCate: Primary category of the mini program

sencondaryCate: Secondary category of the mini program

If both primary and secondary categories are provided, the search results will be the intersection of both categories.

If only one category is provided, the search will be based on that single category.

```
SearchOptions searchOptions = new SearchOptions(keyword, firstCate, sencondaryCate)
```

```
TmfMiniSDK.searchMiniApp(searchOptions, new MiniCallback<List<MiniApp>>() { @Overri
          // Search failed or the mini program list is empty } } });
```

Open the mini program from the search results:

Notes:

The search results are returned via the value method of the MiniCallback API.

The parameter int code indicates the error code.

The parameter String msg indicates the returned search information.

The parameter List<MiniApp> data indicates the list of found mini programs.

```
SearchOptions searchOptions = new SearchOptions(keyword, firstCate, sencondaryCate)

TmfMiniSDK.searchMiniApp(searchOptions, new MiniCallback<List<MiniApp>>() { @Overri
           MiniApp miniApp = data.get(0);
           // Open the first mini program in the search results TmfMiniSDK.startMin
          // Search failed or the mini program list is empty } } });
```

5. Mini program startup listener

To help developers troubleshoot issues, the mini program SDK provides a listener for startup errors. You can configure it with the resultReceiver in MiniStartOptions.

How to configure:

```
private ResultReceiver mResultReceiver = new ResultReceiver(new Handler()) {
    @Override
    protected void onReceiveResult(int resultCode, Bundle resultData) {
      if (resultCode != MiniCode.CODE_OK) {
                String errMsg = resultData.getString("errMsg");
                Toast.makeText(mActivity, errMsg + resultCode, Toast.LENGTH_SHORT
                }
        }
};
```

`The miniStartOptions.resultReceiver` can be used to receive error codes when a mini program is started up. For error code details, see API Description.

Closing Mini Programs

Last updated：2024-11-21 17:26:33

The 'closing mini program' API terminates all activities of a mini program and ends its process.

Closing a specific mini program

API description: This API allows you to close a mini program with a specific ID.

```
/**
 * Terminate a mini program
 *
 * @param context
 * @param appId
 */
public static void stopMiniApp(Context context, String appId)
```

Sample code:

```
TmfMiniSdk.stopMiniApp(context,appId);
```

Closing all mini programs

API description: This API allows you to close all mini programs.

```
/**
 * Terminate all mini programs
 *
 * @param context
 */
public static void stopAllMiniApp(Context context)
```

Sample code:

```
TmfMiniSdk.stopAllMiniApp(context);
```

Deleting Mini Programs

Last updated：2025-01-08 17:39:43

Note：

Running a mini program stores the program package and information locally for faster access in the future. If you want to delete all information of a mini program, you can use the following API to delete a specific mini program or all mini programs.

Deleting a mini program by mini program ID

API description:

```
/**
 * Delete a mini app based on `appId`, regardless of the version (official/develope
 * @param appId
 */
public static void deleteMiniApp(String appId)
```

Sample code:

```
//Delete a mini program with a specific ID
TmfMiniSdk.deleteMiniApp(appId);
```

Deleting a mini program of a specified version type and version number

Delete a mini program of the specified type and the specified version.

API description:

```
/**
* Delete a mini program of the specified type and the specified version
* @param appId
* @param appVerType
* @param version
*/
public static void deleteMiniApp(String appId, int appVerType, String version)
```

Sample code:

Note：

The parameter `appId` is the ID of the mini program.

The parameter 'appVerType' refers to the type of the mini program. You can refer to the mini program type description.

The parameter 'version' is the version number of the mini program.

```
//Delete a mini program with a specific ID, type and version number
TmfMiniSdk.deleteMiniApp(appId, appVerType,version);
```

Delete all mini programs

Delete all cached mini programs.

API description:

```
/**
 * Delete all mini programs (official/developer/preview)
 *
 */
public static void clearMiniAppCache();
```

Sample code:

```
// Delete all mini programs
TmfMiniSdk.clearMiniAppCache();
```

Searching Mini Programs

Last updated：2025-05-26 15:23:20

The mini program SDK provides an API for searching mini programs online by keywords and categories.

API description:

Note:

The SearchOptions parameter is used to specify the keywords and category information for the mini program search.

The MiniCallback parameter is used to obtain the search results of the mini program.

```
/** * Mini program search * * @param searchOptions * @param callback */ public stat
```

Search by keyword

Example:

```
SearchOptions searchOptions = new SearchOptions("yourkeyword"); TmfMiniSDK.searchMi
```

Search by a single category

Example:

```
SearchOptions searchOptions = new SearchOptions("","Category name",""); TmfMiniSDK.
```

Search by dual categories

Note:

The result will be the intersection of the two categories.

Example:

```
SearchOptions searchOptions = new SearchOptions("","Category name","Category name 2
TmfMiniSDK.searchMiniApp(searchOptions, new MiniCallback<List<MiniApp>>() {
    @Override
    public void value(int code, String msg, List<MiniApp> data) {
        if (code == MiniCode.CODE_OK && data != null) {
            // Search successful, list is not empty
        }else{
```

```
            // Search failed, or list is empty
        }
    }
});
```

Search by keyword and categories

Note:

Search by both keyword and categories, and the result will be the intersection of the keyword and category.

Example:

```
SearchOptions searchOptions = new SearchOptions("keyword","Category name","Category
TmfMiniSDK.searchMiniApp(searchOptions, new MiniCallback<List<MiniApp>>() {
    @Override
    public void value(int code, String msg, List<MiniApp> data) {
        if (code == MiniCode.CODE_OK && data != null) {
            // Search successful, list is not empty
        }else{
            // Search failed, or list is empty
        }
    }
});
```

Specify search for mini programs or mini games

Note:

Starting from SDK version 2.2.0, the search API supports specifying whether to search for mini games or mini programs. By default, it searches both.

Example:

```
// Specify search for mini games
SearchOptions searchOptions = new SearchOptions("keyword", "Category name", "Catego
searchOptions.engineType = MiniEngineType.MiniGame;
TmfMiniSDK.searchMiniApp(searchOptions, new MiniCallback<List<MiniApp>>() {
    @Override
    public void value(int code, String msg, List<MiniApp> data) {
        if (code == MiniCode.CODE_OK && data != null) {
            // Search successful, list is not empty
        }else{
            // Search failed, or list is empty
        }
```

```
        }
});
```

Getting the List of Recently Accessed Mini Programs

Last updated：2024-11-21 17:27:40

The mini program SDK provides the following API to get the list of recently accessed mini programs.

```
/**
 * Get the list of recently accessed mini programs
 * @param callback
 */
public static void getRecentList(IRecentMiniCallback callback)
```

Sample code:

```
TmfMiniSDK.getRecentList(new IRecentMiniCallback() {
    @Override
    public void get(List<MiniApp> data) {
        //data is the recently accessed mini programs
    }
});
```

Pre-Downloading Mini Program Package

Last updated：2024-11-21 17:27:58

When a mini program is launched, it synchronizes and updates its version, which may cause a waiting period. To reduce this waiting time and improve user experience, the following API is provided to pre-download mini program packages.

Pre-download a single mini program package

API description:

```
/**
 * Pre-download main package of a mini program
 *
 * @param preDownloadInfo Pre-Downloading Info
 * @param callback Download callback
 */
public static void preDownloadPkg(PreDownloadInfo preDownloadInfo, IDownloadCallbac
```

Sample code:

```
PreDownloadInfo downloadInfo = new PreDownloadInfo("appId");
TmfMiniSDK.preDownloadPkg(downloadInfo, new IDownloadCallback() {
    @Override
    public void onFinish(DownloadInfo downloadInfo) {

    }

    @Override
    public void onError(DownloadInfo downloadInfo) {

    }
});
```

Pre-download multiple mini program main packages

API description:

```
 /**
 * Pre-download main packages of mini programs
```

```
 * @param preDownloadInfos Pre-Downloading Info
 * @param callback Download callback
 */
public static void preDownloadPkg(List<PreDownloadInfo> preDownloadInfos, IDownload
```

Sample code:

```
PreDownloadInfo downloadInfo = new PreDownloadInfo("appId");
PreDownloadInfo downloadInfo2 = new PreDownloadInfo("appId2");
ArrayList<PreDownloadInfo>infos = new ArrayList<>();
infos.add(downloadInfo);
infos.add(downloadInfo2);
TmfMiniSDK.preDownloadPkg(infos, new IDownloadCallback() {
    @Override
    public void onFinish(DownloadInfo downloadInfo) {

    }

    @Override
    public void onError(DownloadInfo downloadInfo) {

    }
});
```

Multi-Process Interaction

Last updated：2024-11-21 17:28:19

The mini program SDK creates a separate process for each mini program, isolating it from the host application's process.
The SDK provides a unified API for communication between the mini program process and the host process.

Check if current process is a mini program process

API description:

```
/**
 * Whether the current process is a mini program process
 * @param context
 * @
 */
public static boolean isMiniProcess(Context context)
```

Host process calls mini program process plugin

1. To implement a sub-process plugin extension

Implementation steps:
Inherit BaseIpcPlugin.
Annotate the class with @IpcMiniPlugin.
Annotate the invoke method with @IpcEvent and define the plugin event name.
Sample code:

```
@IpcMiniPlugin
public class MiniProcessIpcPlugin extends BaseIpcPlugin {
    public static final String EVENT = "MiniProcessPlugin";

    @Override
    @IpcEvent(EVENT)
    public void invoke(IpcRequestEvent req) {

        String value = req.data.getString("key");

        Log.d(ModuleApplet.TAG, "current process:" + ProcessUtil.getProcessName(req
        Bundle resp = new Bundle();
```

```
            resp.putString("key", "i am ok");
            req.ok(resp);
        }
    }
```

2. Plug-ins that call mini program process in main process

API description

```
/**
 * Call a mini-program process plugin for mini program introduction, and this metho
 *
 ///@param appId Mini program ID
 * @param eventId
 * @param request
 * @param callback
 */
public static void callMiniProcessPlugin(String appId, String eventId, Bundle reque

/**
 * Call a mini program process plugin for develop/preview mini program, and this me
 *
 ///@param appId Mini program ID
 * @param appVerType Mini program type. See MiniApp
 * @param eventId
 * @param request
 * @param callback
 */
public static void callMiniProcessPlugin(String appId, int appVerType, String event
```

Sample code:

```
TmfMiniSDK.callMiniProcessPlugin("", MiniProcessIpcPlugin.EVENT, bundle, new IpcCal
    @Override
    public void result(boolean isSucc, Bundle response) {
        Log.d(ModuleApplet.TAG, "current process:" + ProcessUtil.getProcessName(Mod
    }
});
```

Calling host process plugin in mini program process

1. To implement a host process plugin extension

Implementation steps：

Inherit BaseIpcPlugin.

Annotate the class with @IpcMainPlugin.

Annotate the invoke method with @IpcEvent and define the plugin event name.

Sample code:

```
@IpcMainPlugin
public class MainProcessIpcPlugin extends BaseIpcPlugin {
    public static final String EVENT = "MainProcessPlugin";

    @Override
    @IpcEvent(EVENT)
    public void invoke(IpcRequestEvent req) {
        String value = req.data.getString("key");

        Log.d(ModuleApplet.TAG, "current process|" + ProcessUtil.getProcessName(req
        Bundle resp = new Bundle();
        resp.putString("key", "i am ok");
        req.ok(resp);
    }
}
```

2. Calling host process plugin in mini program process

API description

Parameters：

eventId: Indicates event name defined by host process

request: Indicates parameters passed from mini program process to host process plugin

ipcCallBack: Indicates callback to receive the host process

```
/**
 * Calls the main process Plugin, and this method can only be called in the process
 *
 * @param eventId
 * @param request
 * @param callback
 */
public static void callMainProcessPlugin(String eventId, Bundle request, IpcCallbac
```

Sample code:

```
TmfMiniSDK.callMainProcessPlugin(MainProcessIpcPlugin.EVENT, bundle, new IpcCallbac
    @Override
    public void result(boolean isSucc, Bundle response) {
        Log.d(ModuleApplet.TAG, "current process:" + ProcessUtil.getProcessName(Mod
    }
```

```
});
```

Mini Program File Management

Last updated：2024-11-21 17:28:50

The mini program file management API allows for the conversion and creation of file paths between the mini program and the host application's local storage.

Note：

Mini program file path: A path starting with wxfile://, used by mini program developers. The mini program SDK maps these paths to local file paths within the host application.

Host app local file path: An absolute path in the host application's storage, e.g., /data/data/com.tencent.miniapp.demo/app_T1701421723ASSNID/2121/files/mini/.

Get IMiniAppFileManager

All mini program files are managed by IMiniAppFileManager which can be obtained from the mini program context IMiniAppContext. IMiniAppContext can be accessed via BaseJsPlugin.

```
IMiniAppFileManager fileManager = mMiniAppContext.getManager(IMiniAppFileManager.cl
```

CREATING FILES IN THE MINI PROGRAM TEMPORARY DIRECTORY

The SDK supports creating files in the mini program's cache directory and returns the local path for the host app to use.

Sample code:

```
String tmpPath = fileManager.getTmpPath(".jpg");
```

Converting absolute paths to wxfile paths

The SDK can convert file paths created in the mini program's cache directory to paths usable by the mini program.

Sample code:

```
String wxfile = fileManager.getWxFilePath(path);
```

wxfile paths converted to absolute paths

The SDK supports converting wxfile paths in the cache directory to local paths.

Sample code:

```
String path = fileManager.getAbsolutePath(wxfile);
```

Custom Mini Program Capabilities
Custom Mini Program APIs

Last updated：2024-11-21 17:29:23

If a mini program needs to call certain capabilities provided by the host app that are not implemented or cannot be implemented by the mini program SDK, developers can register custom APIs to enable these capabilities.

Implementing custom APIs in the host app

The host app implements the custom mini program API capability by inheriting the BaseJsPlugin.

Note：

Extend `BaseJsPlugin` and define it with the annotation @JsPlugin(secondary = true).

Define a method, which can have only one parameter of the `RequestEvent` type.

Then, define the annotation @JsEvent("event name") in the method. When the mini program JS calls the "event name", the method modified by @JsEvent will be called.

@JsEvent supports defining multiple event names.

The data can be returned synchronously or asynchronously (only one method can be selected for the same event).

You can use `sendState` to send multiple intermediate states to the mini program. After the final sendState call, you must call either ok or fail to indicate the end of the process.

Sample code:

```
@JsPlugin(secondary = true)
public class CustomPlugin extends BaseJsPlugin {
   @JsEvent("customAsyncEvent")
   public void custom(final RequestEvent req) {
       // Get the parameters
       //req.jsonParams
       // Return the data asynchronously
       //req.fail();
       //req.ok();
       JSONObject jsonObject = new JSONObject();
       try {
           jsonObject.put("key", "test");
       } catch (JSONException e) {
           e.printStackTrace();
       }
       req.ok(jsonObject);
    }
```

```
    @JsEvent("customSyncEvent")
    public String custom1(final RequestEvent req) {
        // Get the parameters
        //req.jsonParams
        //Return the data synchronously
        JSONObject jsonObject = new JSONObject();
        try {
            jsonObject.put("key", "value");
        } catch (JSONException e) {
            throw new RuntimeException(e);
        }
        return req.okSync(jsonObject);
     }

    /**
     * Test coverage system API
     * @param req
     */
    @JsEvent("getAppBaseInfo")
    public void getAppBaseInfo(final RequestEvent req) {
        // Get the parameters
        //req.jsonParams
        // Return the data asynchronously
        //req.fail();
        //req.ok();
        JSONObject jsonObject = new JSONObject();
        try {
            jsonObject.put("key", "test");
        } catch (JSONException e) {
            e.printStackTrace();
        }
        req.ok(jsonObject);
     }

    @JsEvent("testState")
    public void testState(final RequestEvent req) {

      try {
          //Call back intermediate state
          req.sendState(req, new JSONObject().put("progress", 1));
          req.sendState(req, new JSONObject().put("progress", 30));
          req.sendState(req, new JSONObject().put("progress", 60));
          req.sendState(req, new JSONObject().put("progress", 100));
      } catch (JSONException e) {
          e.printStackTrace();
      }
```

```
        JSONObject jsonObject = new JSONObject();
        try {
            jsonObject.put("key", "test");
            req.ok(jsonObject);
        } catch (JSONException e) {
          throw new RuntimeException(e);
        }
      }
}
```

Configurecustom APIs

Refer to the following template to configure the custom API parameters:

Note：

In the configuration template, custom API information is stored in a JSON file.

The extApi field in the outer layer of the JSON file holds an array of custom APIs. Each object within extApi represents the configuration of a custom API.

The overwriteWxApi field in the outer layer of the JSON file determines whether custom events override the default mini program API implementation.

Setting this to true means the SDK's built-in mini program APIs will be overridden.

Each custom API configuration must include the following three key attributes:

name: The event name of the custom API, which must match the event name defined by @JsEvent.

sync: Indicates whether the call is synchronous and should be consistent with the return method in the "Creating API" example.

params: Describes the parameters of the custom API.

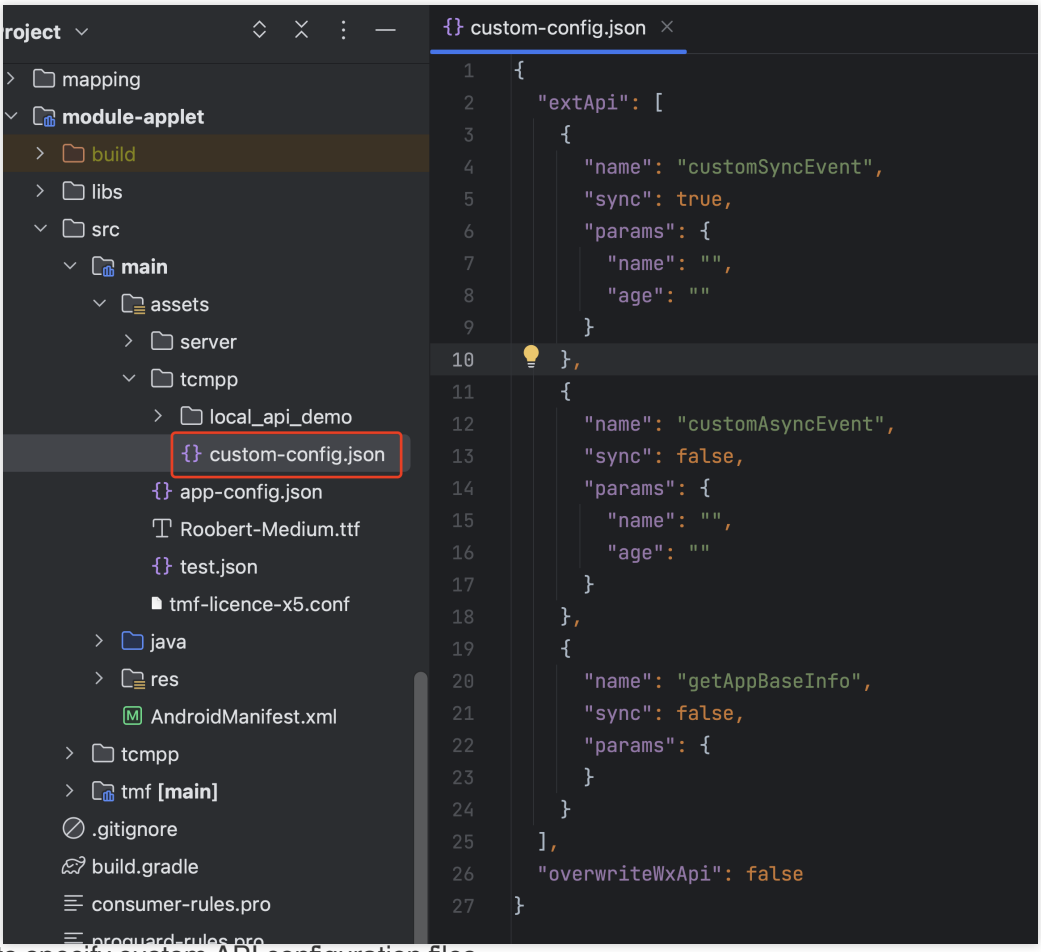
Template：

```
{
 "extApi": [
    {
      //Event name, must match the event defined in the "Create API" example with @
     "name": "customSyncEvent",
     //Whether the call is synchronous, consistent with the return method in the "C
     "sync": true,
     //Define parameters
     //a. JSON format can be nested
     //b. string parameter value can be to ""
     //c.  numeric parameter value can be to 0
     "params": {
       "name": "",
       "age": 0,
```

```
        "object": {
          "key": "",
        }
       }
    },
      {
      "name": "customAsyncEvent2",
      "sync": false,
      "params": {
        "name": "",
        "age": ""
      }
   ],
  //true: If the custom API event name matches a built-in method name in the mini pr
  //API, and the final call will invoke the developer's custom API.
  "overwriteWxApi": false
}
```

Specifying the custom API configuration file path

Place the defined API configuration file in the project's assets directory (the file name and path can be defined by the developer).



Code to specify custom API configuration files

```
@ProxyService(proxy = MiniAppProxy.class)
public class MiniAppProxyImpl extends BaseMiniAppProxyImpl {
     @Override
   public MiniConfigData configData(Context context, int configType, JSONObject par
        if(configType == MiniConfigData.TYPE_CUSTOM_JSAPI) {
            //Custom JsApi configuration
            MiniConfigData.CustomJsApiConfig customJsApiConfig = new MiniConfigData.
            customJsApiConfig.jsApiConfigPath = "tcmpp/custom-config.json";

            return new MiniConfigData
                    .Builder()
                   .customJsApiConfig(customJsApiConfig)
                    .build();
        }

      return null;
}
```

Mini program developer calling custom API

Example

The host app defines two custom APIs: customAsyncEvent and getAppBaseInfo. The configuration file is as follows:

```
{
  "extApi": [
    {
      "name": "customSyncEvent",
      "sync": true,
      "params": {
        "name": "",
        "age": ""
      }
    },
    {
      "name": "customAsyncEvent",
      "sync": false,
      "params": {
        "name": "",
        "age": ""
      }
    },
    {
      "name": "getAppBaseInfo",
      "sync": false,
      "params": {
      }
    }
  "overwriteWxApi": true
}
```

The mini program developers can access custom APIs as follows:

```
wx.customAsyncEvent({"name":"123","age":"18"})
wx.getAppBaseInfo()//his will override the system API and call the custom API.
```

Note：

Since the overwriteWxApi property in the custom API configuration file is set to true, calling wx.getAppBaseInfo in the mini program will invoke the host application's custom implementation of getAppBaseInfo.

Troubleshooting custom APIs

If there are errors when calling custom APIs, see Android FAQs.

Custom API Example

Last updated：2024-11-21 17:29:48

Launching an Activity in a custom API

Start a new Activity in the plugin and retrieve the data returned by the Activity.

Sample code:

Extending BaseJsPlugin to determine the custom API event name:

```
@JsPlugin(secondary = true)
public class CustomPlugin extends BaseJsPlugin {
    @JsEvent("testStartActivityForResult")
    public void testStartActivityForResult(final RequestEvent req) {
  }
}
```

Add a listener for the Activity result and remove it once processing is complete:

Note：

It is recommended to pair the registration and removal of listeners to avoid memory leak.

```
@JsPlugin(secondary = true)
public class CustomPlugin extends BaseJsPlugin {
    @JsEvent("testStartActivityForResult")
    public void testStartActivityForResult(final RequestEvent req) {
      Activity activity = req.activityRef.get();
      TmfMiniSDK.addActivityResultListener(new IActivityResultListener() {
          @Override
          public boolean doOnActivityResult(int requestCode, int resultCode, Intent
              TmfMiniSDK.removeActivityResultListener(this);

              Log.i(ModuleApplet.TAG, data.getStringExtra("key"));
              req.ok();
              return true;
          }
      });
  }
}
```

To start a new activity:

Note：

The requestCode must be >= 1000000 to avoid conflicts with internal request codes, which could cause unexpected issues.

```
@JsPlugin(secondary = true)
public class CustomPlugin extends BaseJsPlugin {
    @JsEvent("testStartActivityForResult")
    public void testStartActivityForResult(final RequestEvent req) {
      Activity activity = req.activityRef.get();
      TmfMiniSDK.addActivityResultListener(new IActivityResultListener() {
          @Override
          public boolean doOnActivityResult(int requestCode, int resultCode, Intent
              TmfMiniSDK.removeActivityResultListener(this);

              Log.i(ModuleApplet.TAG, data.getStringExtra("key"));
              req.ok();
              return true;
          }
      });

      //Note: requestCode must be >= 1000000 to avoid conflicts with internal reque
      activity.startActivityForResult(new Intent(activity, TransActivity.class), 10
  }
}
```

Calling third-party apps in custom APIs

When calling other third-party apps (e.g., for sharing, payment, login) within a custom API, ensure that the operation returns directly to the mini program instead of the host app.

Firstly, developers create transparent auxiliary Activity.

Define a transparent helper activity, such as TransActivity. The transparency settings depend on the theme configuration, which is related to the system Activity that your app's Activity inherits from.

```
  <activity android:name=".activity.TransActivity"
    android:exported="false"
    android:screenOrientation="portrait"
    android:theme="@style/TransparentTheme"/>
```

And then invoke that business logic in the TransActivity.

```
public class TransActivity extends AppCompatActivity {
   @Override
   protected void onCreate(@Nullable Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.applet_activity_tran);

       /************************
```

```
                Business development
        ************************/

        //After completing the business logic, return data and call finish
        Intent intent = new Intent();
        intent.putExtra("key", "value");
        setResult(RESULT_OK, intent);
        finish();
    }
}
```

The plug-in starts the activity and listens for the data to return.

```
  @JsPlugin(secondary = true)
public class CustomPlugin extends BaseJsPlugin {
    @JsEvent("testStartActivityForResult")
    public void testStartActivityForResult(final RequestEvent req) {
      Activity activity = req.activityRef.get();
      TmfMiniSDK.addActivityResultListener(new IActivityResultListener() {
          @Override
          public boolean doOnActivityResult(int requestCode, int resultCode, Intent
              TmfMiniSDK.removeActivityResultListener(this);

              Log.i(ModuleApplet.TAG, data.getStringExtra("key"));
              req.ok();
              return true;
          }
      });

      //Note: requestCode must be >= 1000000 to avoid conflicts with internal reque
      activity.startActivityForResult(new Intent(activity, TransActivity.class), 10
  }
}
```

Custom Mini Program View Components

Last updated：2025-05-26 15:23:48

Mini program view component extension

Users of mini program SDK can customize client native components for the mini program. Mini program developers can create and operate native components on mini program pages and communicate with them by using specific mini program APIs. On Android, native client components are rendered above the mini-program page, do not support zIndex, and will always overlay other mini-program components, potentially obscuring mini-program content. These components are drawn as regular Views.

Notes:

To implement extended components, both the mini program developers and the host application developers need to make modifications. The combination of these modifications enables the extension of mini program view components.

Implementation of native component extensions

App developers need to implement specific proxies to create native components and respond to operations on them when the mini program calls custom component APIs.

Native component extensions

To customize the creation of native components, the host can override the proxy settings as follows:

```
@ProxyService(proxy = ExternalElementProxy.class)
public class MyExternalElementProxy extends ExternalElementProxy{}
```

The proxy needs to implement the following methods:

1. Insert a component. This method is called when the mini program inserts a native component into the page. Developers need to implement this method and add the custom component as a child View to the container provided by the parent parameter.

```
/** * Creates a component. This API is called when the mini program creates a custo
```

2. Update a component. This method is called to notify the app when the position or size of the native component changes within the mini program. The layout of the parent container of the native component will be adapted to the new style, and the native component itself can be adjusted as needed.

```
/** * Updates component style. This API is called when the mini program updates the
```

3. Operate a component. This method is called when the mini program sends an event to a native component (e.g., button click, status change). The specific content of the event needs to be defined by the developer in params.

```
/** * Operates a component. This API is called when the mini program needs to send
```

4. Delete a component. This method is called to notify the app when the mini program deletes a native component. At this point, the component's parent container is removed and the app should destroy the component.

```
/** * Deletes a component. * * @param widgetId The ID of the mini program component
```

Mini program native component context

The context of mini program components contains methods that allow native components to send messages to the corresponding mini program components. The onExternalElementEvent method is used to send an onExternalElementEvent event directly to the mini program, and the mini program should capture and process the event. The callbackSuccess and callbackFail methods are used to call the success or fail callbacks respectively when the mini program calls an API to send an event to the app.

```
/** * Sends an onExternalElementEvent event to the mini program.* * @param jsonObje
```

Implementation of mini program extension

To insert a custom client-side component into a mini program page, you need to include an external-element node in the WXML:

```
<external-element
    id="comp1"
    type="maTestView"
    _insert2WebLayer="true"
    style="width: 200px;height: 100px;"
    bindexternalelementevent="handleEvent"
></external-element>
```

The type should match the component type name agreed upon with the app. `_insert2WebLayer` indicates whether the component is a same-layer component or a non-same-layer component (true for same-layer, which requires the client to implement a same-layer component proxy; false for non-same-layer, which requires the client to implement a non-same-layer proxy). The bindexternalelementevent can capture events passed from the native side, such as onExternalElementEvent or onXWebExternalElementEvent. The callback parameters include:

```
{
    target,
    currentTarget,
```

```
    timeStamp,
    touches,
    detail, // Parameters passed by native
}
```

Then, the mini program creates a context associated with this node using its ID.

```
this.ctx = wx.createExternalElementContext('comp1');
```

This method notifies the application to create the corresponding native component at the position of the node. Once the context is created, the mini program can send events to the native component and perform operations on it through this context:

```
this.ctx.call({
  params1: {
    name: 'name1',
    age: 11
  },
  params2: {
    name: 'name2',
    age: 22
  },
  success: (e) => {
    console.log('====operate success=====', e)
  },
  fail: (e) => {
    console.log('====operate fail=====', e)
  },
  complete: (e) => {
    console.log('====operate complete=====', e)
  }
})
```

Custom SDK Capabilities
Custom Mini Program UI

Last updated：2025-01-23 17:22:59

The mini program SDK now allows developers to customize certain native UI effects of mini programs. Follow the guidelines below to configure this feature.

To do this, you need to extend and implement AbsMiniUiProxy. Below is an example:

```
@ProxyService(proxy = IMiniUiProxy.class)
public class MiniUiProxyImpl extends AbsMiniUiProxy
```

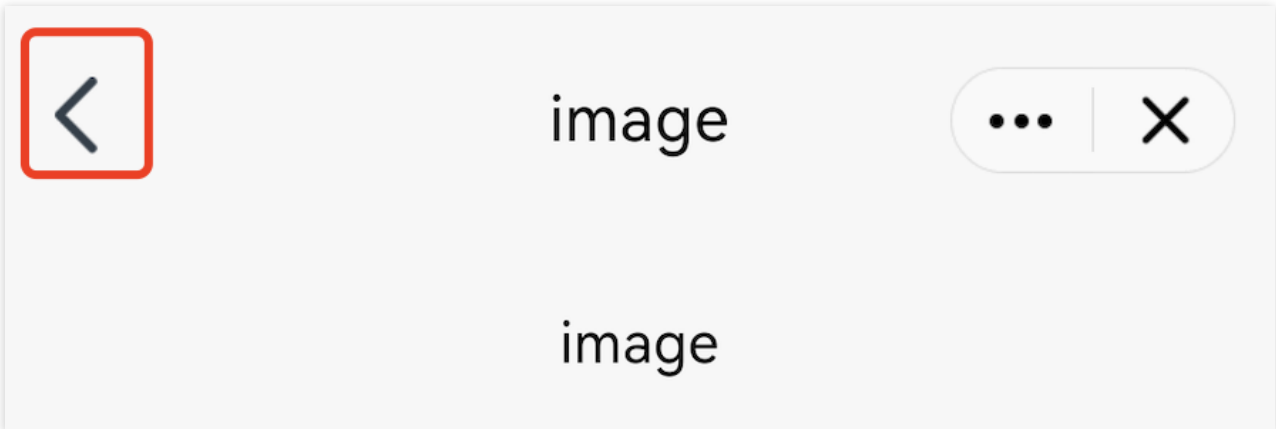
Note:

You must use `@ProxyService(proxy = IMiniUiProxy.class)` annotation to customize the UI.

Customize the navigation bar button

1. Back button

The default style of the Back button is shown in the following figure:



The mini program SDK allows you to customize the Back button by **overriding the navBarBackRes method of AbsMiniUiProxy.**

The API is defined as follows:

Method parameter mode: The color of the navigation bar title. Valid values: 1 (black), 0 (white).

Method return value: The icon resource ID of the Back button.

```
/**
 * Customizes the navigation bar back button icon. The required size is 24 x 43 pix
 * Called in the subprocess.
 *
```

```
     * @param mode Navigation bar title color. Valid values: 1 (black), 0 (white).
     * @return
     */
    @DrawableRes
    int navBarBackRes(int mode);
```

Note:

The size of the Back button icon resource is required to be 24 x 43 pixels.

Example:

```
@Override public int navBarBackRes(int mode) { if(mode == 0) {//black return R.draw
```

2. Home button

The default style of the Home button is shown in the following figure:

The mini program SDK allows you to customize the Home button by **overriding the homeButtonRes method of AbsMiniUiProxy**.

The API is defined as follows:

Method parameter mode: The color of the navigation bar title. Valid values: 1 (black), 0 (white).

Method return value: The icon resource ID of the Home button.

```
/**
 * Customize the Home button icon on the navigation bar. The required size is 48 x
 * Called in the subprocess.
 *
 * @param mode Navigation bar title color. Valid values: 1 (black), 0 (white).
 * @return
 */
@DrawableRes
int homeButtonRes(int mode);
```

Note:

The size of the Home icon on the navigation bar is required to be 48 x 48 pixels.

Example:

```
@Override public int homeButtonRes(int mode) { if(mode == 0) {//black return R.draw
```

3. More button

The default style of the More button is shown in the following figure:

The mini program SDK allows you to customize the More button by **overriding the moreButtonRes method of AbsMiniUiProxy.**

The API is defined as follows:

Method parameter mode: The color of the navigation bar title. Valid values: 1 (black), 0 (white).

Method return value: The icon resource ID of the More button.

```
/**
 * Customizes the More button icon on the navigation bar.The required size is 80 x
 * Called in the subprocess.
 *
 * @param mode Navigation bar title color. Valid values: 1 (black), 0 (white).
 * @return
 */
@DrawableRes
int moreButtonRes(int mode);
```

Note:

The size of the More button icon on the navigation bar is required to be 80 x 59 pixels.

Example:

```
@Override public int moreButtonRes(int mode) { if(mode == 0) {//black return R.draw
```

4. Close button

The default style of the Close button is shown in the following figure:

The mini program SDK allows you to customize the Close button by the following method.

Customize the Close button:

Override the closeButtonRes method of AbsMiniUiProxy.

The API is defined as follows:

API description:

Method parameter mode: The color of the navigation bar title. Valid values: 1 (black), 0 (white).

Method return value: The icon resource ID of the Close button.

```
/**
 * Customize the Close button on the navigation bar. The required size is 80 x 59 p
 * Called in the subprocess.
 *
 * @param mode Navigation bar title color. Valid values: 1 (black), 0 (white).
 * @return
 */
@DrawableRes
int closeButtonRes(int mode);
```

Note:

The size of the Close button icon on the navigation bar is required to be 80 x 59 pixels.

Example:

```
@Override public int closeButtonRes(int mode) { if(mode == 0) {//black return R.dra
```

5. The vertical bar in the middle of the capsule button

The default style of the vertical bar in the middle of the capsule button is shown in the following figure:

The mini program SDK allows you to customize the vertical bar in the middle of the capsule button by **overriding the lineSplitBackgroundColor method of AbsMiniUiProxy.**

The API is defined as follows:

Method return value:The background color of the vertical bar.

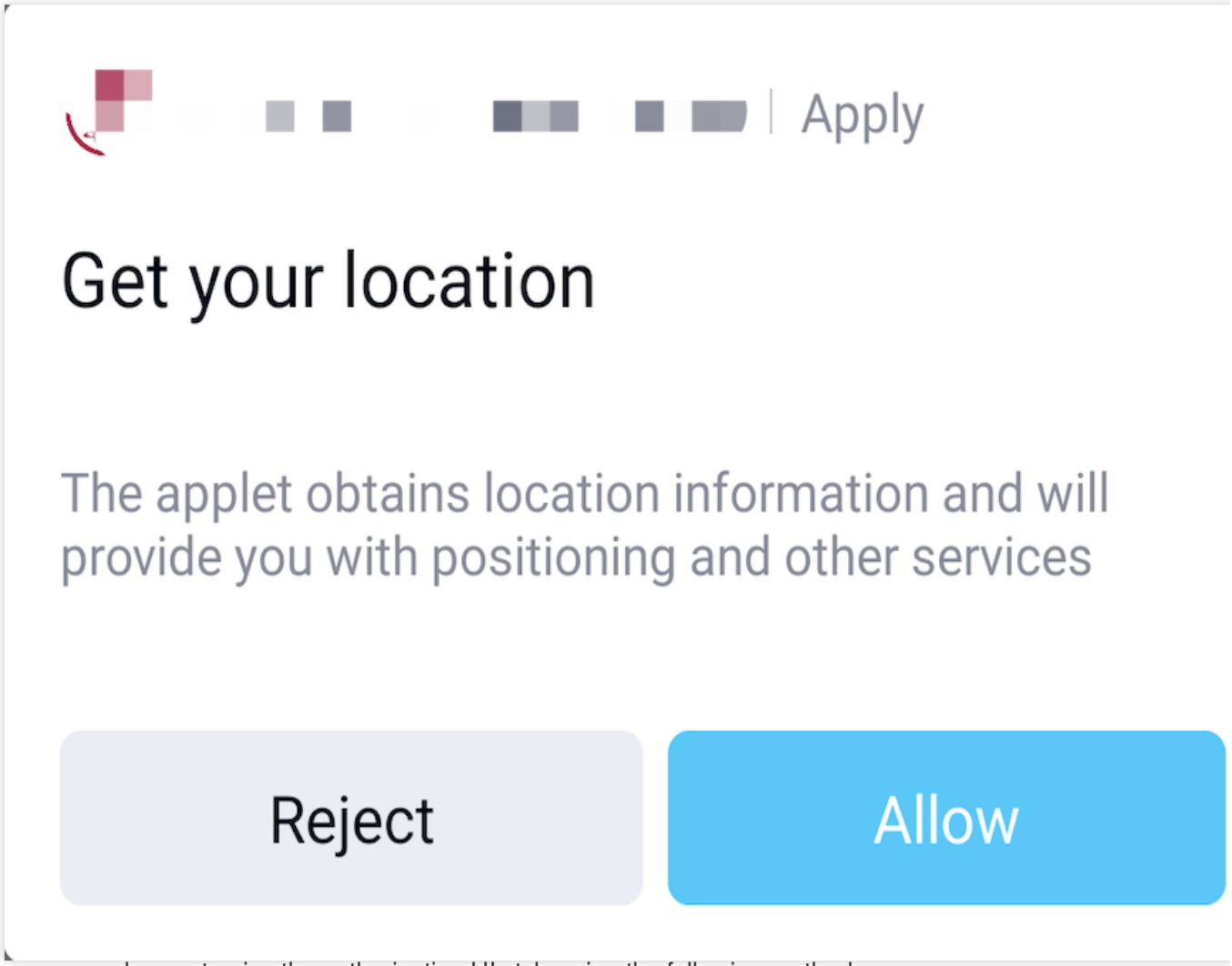
```
/**
 * The background color of the vertical bar in the middle of the capsule button.
 * Called in the subprocess.
 *
 * @return
 */
@DrawableRes
int lineSplitBackgroundColor();
```

Example:

```
@Override public int lineSplitBackgroundColor() {
return Color.RED; }
```

Custom authorization popup

When a mini program calls an API that requires authorization, the SDK provides a default authorization UI style.



Developers can also customize the authorization UI style using the following method.

Customize authorization popup:

Override the authView method of AbsMiniUiProxy.

The parameter Context indicates the context of the mini program process.

The parameter MiniAuthInfo contains the authorization information requested by the current mini program, see MiniAuthInfo for details.

The parameter IAuthView is used to set the custom popup view.

The API is defined as follows:

```
/**
 * Customizes the custom popup view.
 * Called in the subprocess.
 *
 * @param context
 * @param authInfo
 * @param authView
 * @return true: Use the custom authorization view; false: Use the default view.
 */
@Override
```

```
public boolean authView(Context context, MiniAuthInfo authInfo, IAuthView authView)
```

Notes:

The return value of the authView method indicates whether to use a custom authorization view. The false indicates using the default authorization view, while the true indicates using a custom authorization view.

The refuseListener and grantListener in MiniAuthInfo must be set, otherwise, the mini program authorization will fail.

Below is an example:

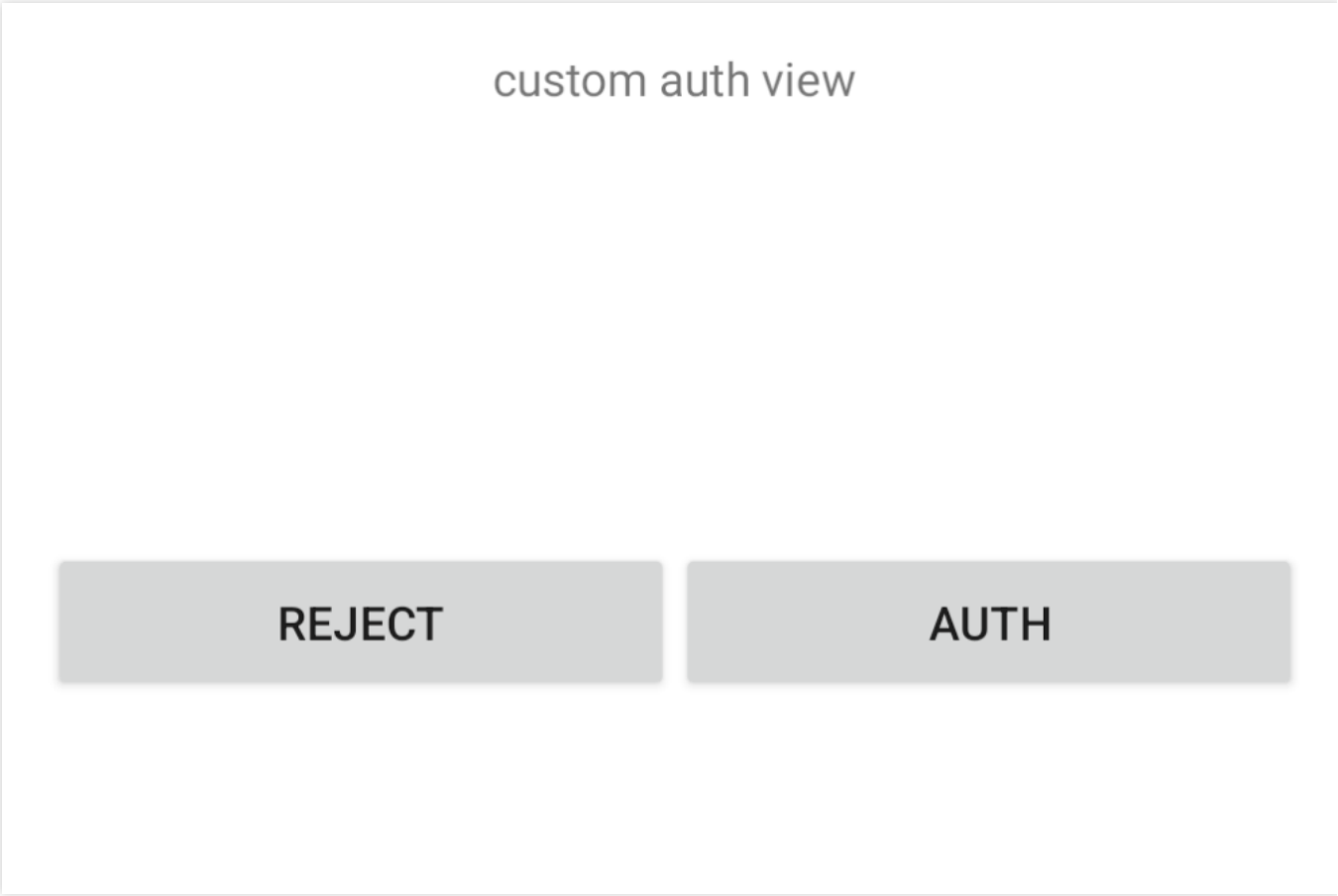
View layout file

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingTop="16dp"
    android:paddingRight="16dp"
    android:paddingBottom="15dp"
    android:background="@android:color/white"> <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="custom auth view"
        android:layout_centerHorizontal="true"/> <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="150dp"> <Button
            android:id="@+id/mini_auth_btn_refuse"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:text="@string/applet_mini_reject" /> <Button
            android:id="@+id/mini_auth_btn_grant"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:text="@string/applet_mini_auth" /> </LinearLayout> </RelativeLa
```

Override the authView method (note that the return value is true):

```
@Override public boolean authView(Context context, MiniAuthInfo authInfo, IAuthView
view.findViewById(R.id.mini_auth_btn_refuse).setOnClickListener(authInfo.refuseList
view.findViewById(R.id.mini_auth_btn_grant).setOnClickListener(authInfo.grantListen
authView.getView(view); } return isCustom; }
```

Custom authorization view:

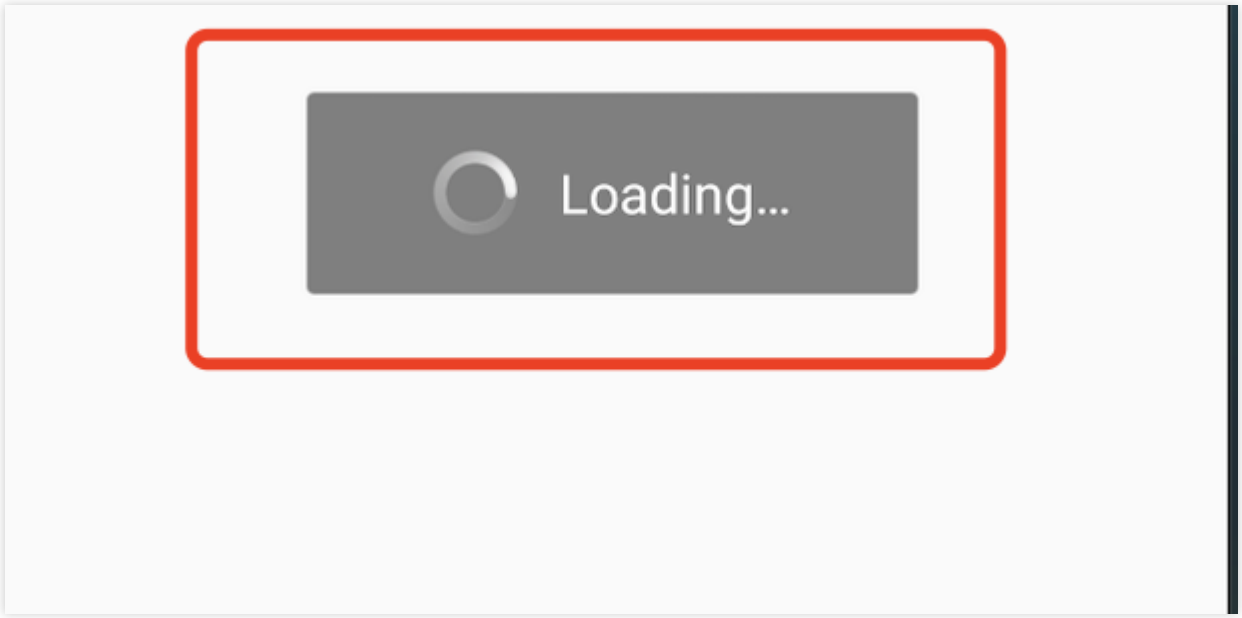


Customize the mini program loading views

Mini programs have loading animations for checking updates and starting up, both of which can be customized.

1. Customize update loading view

The default update loading animation is shown below:



You can **customize it by overriding the updateLoadingView method of AbsMiniUiProxy.**

API description:

Notes:

The return value of the updateLoadingView method is an instance of the IMiniLoading.

IMiniLoading methods:

Create: Creates a loading view.

Show: Callback when showing the loading effect.

Stop: Callback when stopping the loading effect.

```
/**
 * Customizes the loading page for checking a mini program's updates.
 * Called in the main process.
 *
 * @param context
 * @return
 */
public abstract IMiniLoading updateLoadingView(Context context);
```

Example:

```
@Override
public IMiniLoading updateLoadingView(Context context) {
    return new IMiniLoading() {
        @Override
        public View create() {
            return LayoutInflater.from(context).inflate(R.layout.applet_activity_cu
        }
```

```
        @Override
        public void show(View v) {

        }

        @Override
        public void stop(View v) {

        }
    };
}
```

2. Customize the startup loading view

You can customize the startup loading view for mini program by **overriding the startLoadingView method of AbsMiniUiProxy.**

API description:

Notes:

The return value of the startLoadingView method is an instance of the IMiniLoading.

IMiniLoading methods:

Create: Creates a loading view.

Show: Callback when showing the loading effect.

Stop: Callback when stopping the loading effect.

```
/**
 * Customizes the startup loading page for mini program.
 * Called in the subprocess.
 *
 * @param activityWeakRef Activity reference
 * @param app Mini program information
 * @return Returns the mini program loading UI
 */
public abstract IMiniLoading startLoadingView(WeakReference<Activity> activityWeakR
```

Example:

```
@Override
public IMiniLoading startLoadingView(Context context) {
    return new IMiniLoading() {
```

```
        @Override
        public View create() {
            return LayoutInflater.from(context).inflate(R.layout.applet_activity_cu
        }

        @Override
        public void show(View v) {

        }

        @Override
        public void stop(View v) {

        }
    };
}
```

Hide the capsule button on the loading page

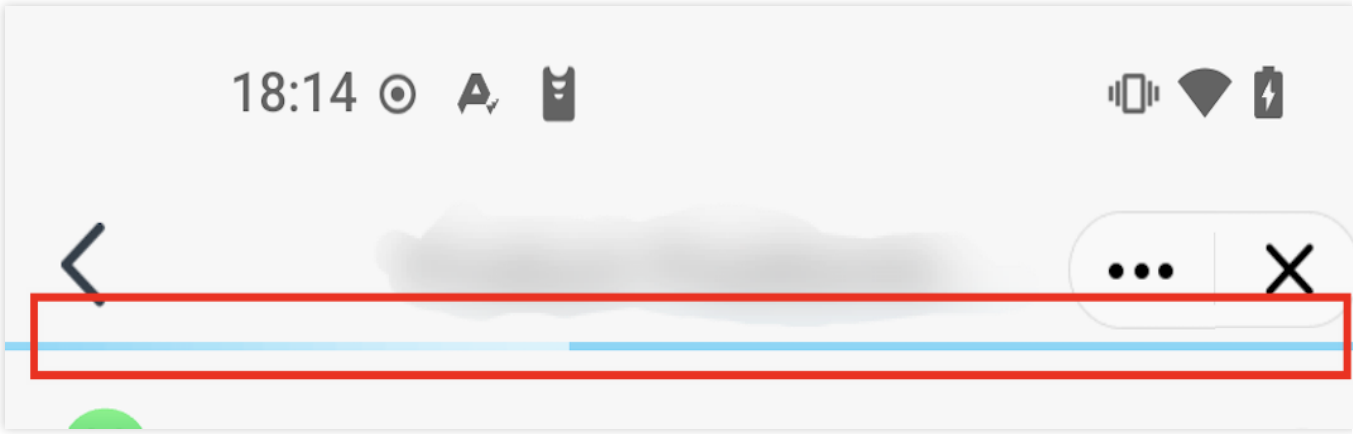
A capsule button is shown in the upper right corner of the mini program loading page by default.**You can choose to hide/show the capsule button by overriding the hideLoadingCapsule method of AbsMiniUiProxy.**
The capsule button is shown in the following figure:

API description: The return value of true indicates that the capsule button on the loading page should be hidden, while false (the default) indicates that it should be shown.

```
/** * Whether to hide the capsule button in the upper right corner of the mini prog
```

Customize the web-view component progress bar

When loading a page, the web-view component displays a progress bar by default.



The SDK provides customization options for this progress bar.

1. Hide the web-view component progress bar

Override the hideWebViewProgressBar method of AbsMiniUiProxy to set whether to hide the progress bar

```
/**
 * Whether to hide the web-view component progress bar
 * @return true: Hide; false: Show (the default)
 */
boolean hideWebViewProgressBar();
```

2. Set the color of the web-view component progress bar

Override the webviewProgressBarColor and webviewProgressBarBgColor methods of AbsMiniUiProxy to set the color of the progress bar

```
/**
 * Sets the color of the progress bar (color of the completed progress). Returns nu
 * @return Progress bar color value, for example: Color.GREEN
```

```
     */
    Integer webviewProgressBarColor();

    /**
     * The background color of the progress bar (the color of unfinished progress). Ret
     * @return Background color value of the progress bar. For example: Color.GREEN
     */
    Integer webviewProgressBarBgColor();
```

Custom Mini Program SDK APIs

Last updated：2025-04-28 18:13:11

To enhance usage scenarios, the mini program SDK exposes certain APIs that can be customized by the host app. This allows the host app to provide unique capabilities to the mini program.

Prerequisite

To customize the mini program SDK APIs, you need to define a custom MiniAppProxy as follows:

Note：

Define an implementation class that extends BaseMiniAppProxyImpl and annotate it with @ProxyService(proxy = MiniAppProxy.class).

```
@ProxyService(proxy = MiniAppProxy.class)
public class MiniAppProxyImpl extends BaseMiniAppProxyImpl{}
```

Once the prerequisites are completed, you can customize the mini program SDK APIs. The SDK mainly supports the following types of APIs:

1. Customizing capsule button feature

1.1 Customizing close button event listener

Customize the close button event listener for the capsule button, allowing the host app to receive callback events when the close button is clicked.

Close button figure:

API description: The return value of onCapsuleButtonCloseClick method indicates whether to customise the close button listener or not, a return value of true means customise it, false (default value) means use the default listener.

```
/**
 * Close option for clicking capsule button
 * Calling environment: child process
 *
 * @param miniAppContext The environment in which the mini programme runs (the mini
 * @param onCloseClickedListener Callback when clicking the mini program to close i
 * @return does not support this interface, please return false
 */
public abstract boolean onCapsuleButtonCloseClick(IMiniAppContext miniAppContext,
```

```
        DialogInterface.OnClickListener onCloseClickedListener);
```

Sample code:

```
    @Override
    public boolean onCapsuleButtonCloseClick(IMiniAppContext miniAppContext,
                                             DialogInterface.OnClickListener onClos
        Log.e("TAG","onCapsuleButtonCloseClick"+miniAppContext.getMiniAppInfo());
        //handle close mini app event
        return true;
    }
```

1.2 Customising the More Buttons event listener

Customising the event listener of the More button allows the host application to listen to the callback event of the corresponding item when the More button is clicked.
Diagram of the More button:

API description: The getMoreItemSelectedListener method returns a value that indicates more button listeners.

```
/**
 * Return to Capsule More panel button click listener
 * Calling environment: child process
 *
 * @return listener
 */
public abstract OnMoreItemSelectedListener getMoreItemSelectedListener();
```

Sample code:

```
/**
 * Return to Capsule More panel button click listener
 * @return
 * @return
 */
@Override
public OnMoreItemSelectedListener getMoreItemSelectedListener() {
    return new DemoMoreItemSelectedListener();
}

/**
define of DemoMoreItemSelectedListener
```

```
**/
public class DemoMoreItemSelectedListener extends DefaultMoreItemSelectedListener {
    public static final int CLOSE_MINI_APP = 150;

    @Override
    public void onMoreItemSelected(IMiniAppContext miniAppContext, int moreItemId)

        switch (moreItemId) {
            case CLOSE_MINI_APP:
                close(miniAppContext);
                return;
            case OTHER_MORE_ITEM_1:
                miniAppContext.getAttachedActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(miniAppContext.getAttachedActivity(), "custo
                    }
                });
                return;
        }
        super.onMoreItemSelected(miniAppContext, moreItemId);
    }

    public void close(IMiniAppContext miniAppContext) {
        Activity activity = miniAppContext.getAttachedActivity();
        if (activity != null && !activity.isFinishing()) {
            boolean moved = activity.moveTaskToBack(true);
            if (!moved) {
                QMLog.e("Demo", "moveTaskToBack failed, finish the activity.");
                activity.finish();
            }
        }
    }
}
```

1.3 Customising the More Buttons Display List

By the time the user triggers the more buttons click event, the following list of optional extended buttons will pop up, the default list is shown below:

By overriding MiniAppProxy's getMoreItems method, it is possible to customise the list of buttons for more menu extensions.

API Description:

The return value of getMoreItems method represents the customised list of extension buttons;

The method parameter MoreItemList.Builder is used to add extension buttons, the display order of extension buttons is the same as the add order;

The id attribute of MoreItem is used to distinguish the buttons and needs to be unique.

```
/**
 * The button that returns the capsule to the More panel, the ID of the extended bu
 * Call environment: child process
 *
 * @param builder
 * @return
 */
public abstract ArrayList<MoreItem> getMoreItems(MoreItemList.Builder builder);
```

Warning：

The ID of the extension button needs to be set to a value in the interval [100, 200], otherwise the add is invalid.

Sample code:

```
@Override
public ArrayList<MoreItem> getMoreItems(IMiniAppContext miniAppContext, MoreItemLis
    MoreItem item1 = new MoreItem();
    item1.id = ShareProxyImpl.OTHER_MORE_ITEM_1;
    item1.text = getString(miniAppContext, R.string.applet_mini_proxy_impl_other1);
    item1.drawable = R.mipmap.mini_demo_about;

    MoreItem item2 = new MoreItem();
    item2.id = ShareProxyImpl.OTHER_MORE_ITEM_2;
    item2.text = getString(miniAppContext, R.string.applet_mini_proxy_impl_other2);
    item2.shareKey = SHARE_TWITTER;//Custom sharing key, must be set and unique, wi
    item2.drawable = R.mipmap.mini_demo_about;

    MoreItem item3 = new MoreItem();
    item3.id = DemoMoreItemSelectedListener.CLOSE_MINI_APP;
    item3.text = getString(miniAppContext, R.string.applet_mini_proxy_impl_float_ap
    item3.drawable = R.mipmap.mini_demo_about;

    MoreItem item4 = new MoreItem();
    item4.id = ShareProxyImpl.OTHER_MORE_ITEM_INVALID;
    item4.text = getString(miniAppContext, R.string.applet_mini_proxy_impl_out_of_e
    item4.drawable = R.mipmap.mini_demo_about;

    // Self-adjusting order.
    builder.addMoreItem(item1)
            .addMoreItem(item2)
            .addShareQQ("QQ", R.mipmap.mini_demo_channel_qq)
            .addMoreItem(item3)
```

```
            .addShareQzone(getString(miniAppContext, R.string.applet_mini_proxy_imp
                    R.mipmap.mini_demo_channel_qzone)
            .addShareWxFriends(getString(miniAppContext, R.string.applet_mini_proxy
                    R.mipmap.mini_demo_channel_wx_friend)
            .addShareWxMoments(getString(miniAppContext, R.string.applet_mini_proxy
                    R.mipmap.mini_demo_channel_wx_moment)
            .addRestart(getString(miniAppContext, R.string.applet_mini_proxy_impl_r
                    R.mipmap.mini_demo_restart_miniapp)
            .addAbout(getString(miniAppContext, R.string.applet_mini_proxy_impl_abo
                    R.mipmap.mini_demo_about)
            .addDebug(getString(miniAppContext, R.string.mini_sdk_more_item_debug),
                    R.mipmap.mini_demo_about)
            .addMonitor(getString(miniAppContext, R.string.applet_mini_proxy_impl_p
                    R.mipmap.mini_demo_about)
            .addComplaint(getString(miniAppContext, R.string.applet_mini_proxy_impl
                    R.mipmap.mini_demo_browser_report)
            .addSetting(getString(miniAppContext, R.string.mini_sdk_more_item_setti
                    R.mipmap.mini_demo_setting);

    return builder.build();
}

private String getString(IMiniAppContext miniAppContext, int id) {
    return miniAppContext.getContext().getString(id);
}
```

2. Customised Image Selection

The mini program SDK provides default image selection, which is done by default using the album selector that comes with Android, and is triggered by calling multimedia selection wx.chooseImage in the mini program.
Custom image selection can be achieved by overriding the openChoosePhotoActivity method of BaseMiniAppProxyImpl.

API description.

Parameter Context: the current activity instance of the mini program;

Parameter maxSelectedNum: maximum number of selected pictures;

Parameter IChoosePhotoListner: callback interface for photo selection;

Return value boolean: a return value of false means using the default image selection implementation, a return value of true means using the customised image selection implementation.

```
/**
 * Open the selection screen
 *
```

```
 * @param context Current Activity
 * @param maxSelectedNum Maximum number of allowable selections
 * @param listner Callback interface
 * @return Do not support this interface, please return false
 */
@Override
public boolean openChoosePhotoActivity(Context context, int maxSelectedNum, IChoose
```

Note：

When the image path is returned by the onResult method of IChoosePhotoListner, the image path should be an absolute path.

Sample code:

```
@Override
public boolean openChoosePhotoActivity(Context context, int maxSelectedNum, IChoose
    Log.d("TAG","open choose photo activity ");
    Intent intent = new Intent(context, ChoosePhotosActivity.class);
    intent.putExtra("maxCount",maxSelectedNum);
    //not recommand to use static refs,just for example
    ChoosePhotosActivity.setChooseCallBack(listner);
    context.startActivity(intent);

    return true;
}


public class ChoosePhotosActivity extends Activity {
   static WeakReference<MiniAppProxy.IChoosePhotoListner>iChoosePhotoListnerWeakRef
    public static void setChooseCallBack(MiniAppProxy.IChoosePhotoListner choosePho
        iChoosePhotoListnerWeakReference = new WeakReference<>(choosePhotoListner);
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //todo show your choose view
    }

    @Override
    protected void onResume() {
        super.onResume();
        startChooseImage();
    }

    private void startChooseImage(){
        //todo finish choose logic
```

```
        //response choose callback
        ArrayList<String>path = new ArrayList<>();
        //todo replace with real path of image
        path.add("picture local absolute path");
        path.add("picture local absolute path2");
        path.add("/data/user/0/com.tencent.tmf.miniapp.demo/files/userfiles/Screens
        iChoosePhotoListnerWeakReference.get().onResult(path);
    }
}
```

3. Customising image preview

The mini program SDK provides a default image preview implementation, which is triggered when the mini program calls the multimedia selection wx.previewImage.
Custom image preview can be achieved by overriding the openImagePreview method of BaseMiniAppProxyImpl.
The default image preview page is shown below:

API description:
Parameter Context: the current activity instance of the mini program;
parameter selectedIndex: position of the selected image in the list;
parameter pathList: list of paths of the images to be displayed;
Return value boolean: a return value of false means use default image to display, a return value of true means use custom image to display.

```
/**
 * Open the picture preview interface
 * Calling environment: child process
 *
 * @param context Current Activity
 * @param selectedIndex index of the currently selected image
 * @param pathList picture path list
 * @return Do not support this interface, please return false
 */
public abstract boolean openImagePreview(Context context, int selectedIndex, List<S
```

Sample code:

```
/**
 * Open the picture preview interface
 *
 * @param context Current Activity
```

```
 * @param selectedIndex index of the currently selected image
 * @param pathList picture path list
 * @return Do not support this interface, please return false
 */
@Override
public boolean openImagePreview(Context context, int selectedIndex, List<String> pa
    //todo start your image preview
    Intent intent = new Intent(context, CustomPreviewActivity.class);
    intent.putExtra("curIndex",selectedIndex);
    intent.putStringArrayListExtra("pathList", pathList);
    context.startActivity(intent);
    return true;
}
```

4. Customising the user information in the authorisation pop-up

When a mini program applies for user information, it will pop up the user authorisation pop-up window; the host application can customize part of the information in the user authorisation pop-up window, including: avatar image and user nickname, as shown in the following figure:

Customisation can be achieved by overriding the getUserInfo method of BaseMiniAppProxyImpl.

API description:

Parameter appId: appId of the mini program currently requesting authorisation;

Parameter AsyncResult: used to return user information to the mini program SDK.

```
/**
 * Get scope.userInfo authorised user information
 * Call environment: subprocess
 *
 * @param appId
 * @param result
 */
@Override
public void getUserInfo(String appId, AsyncResult result)
```

Sample code:

```
/**
 * Get scope.userInfo authorised user information
 * Call environment: subprocess
 *
 * @param appId
 * @param result
```

```
 */
@Override
public void getUserInfo(String appId, AsyncResult result) {
    JSONObject jsonObject = new JSONObject();
    try {
        //return nickname
        jsonObject.put("nickName", "userInfo Test");;
        //return the avatar url
        jsonObject.put("avatarUrl", "https://gimg2.baidu.com/image_search/src=http%
        result.onReceiveResult(true, jsonObject);
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

5. Custom Image Loader

Warning:

Since the mini program SDK does not have a default image loader implementation, the loading of images relies on the host application's implementation; if the host application does not implement an icon loader, this will result in abnormal image loading on some pages.

Host app developers, need to implement image loading customisation by overriding the getDrawable method of BaseMiniAppProxyImpl.

API description:

Parameter context: the context of the mini program process currently requesting authorisation;

parameter source: the source of the image, it can be a local or network image;

parameter width: width of the image;

parameter height: height of the image;

parameter defaultDrawable: the default drawable, used in loading and loading failure;

Return value Drawable: the loaded drawable object.

```
@Override
public Drawable getDrawable(Context context, String source, int width, int hight, D
```

Sample code:

```
@Override
public Drawable getDrawable(Context context, String source, int width, int hight, D
    //Access to access their own ImageLoader
    //sample code to use the open source universalimageloader
    UniversalDrawable drawable = new UniversalDrawable();
    if (TextUtils.isEmpty(source)) {
```

```
            return drawable;
        }
        drawable.loadImage(context, source);
        return drawable;
    }
```

6. Custom mini program data storage segregated by account

Override the getAccount method of BaseMiniAppProxyImpl to achieve isolated storage of mini program data.
API description: The return value is the user account (must be unique), the data will be stored in isolation according to the account after setting, it is not recommended to use user sensitive information.

```
/**
 * user account, must be unique, after setting the data will be stored according to
 * Call environment: the main process
 */
@Override
public String getAccount() {
    return "tmf_test";
}
```

7. Customised Virtual Domains

Mini program to play to the default virtual domain name, this domain name is not a real domain name, in the browser is not accessible, if you need to customise can be set up in accordance with the following configuration.
The default virtual domain name is shown below:

To customise the virtual domain name, you need to override the configData method of BaseMiniAppProxyImpl and intercept the configType to MiniConfigData.TYPE_DOMAIN to achieve customisation. Mini program to play to the default virtual domain name, this domain name is not a real domain name, in the browser is not accessible, if you need to customise can be set up in accordance with the following configuration.
The default virtual domain name is shown below:
Sample code:

```
@Override
public MiniConfigData configData(Context context, int configType, JSONObject params
    if(configType == MiniConfigData.TYPE_DOMAIN) {
        //Virtual Domain Configuration
        MiniConfigData.DomainConfig domainConfig = new MiniConfigData.DomainConfig(
```

```
        domainConfig.domain = "test.com";

        return new MiniConfigData
                .Builder()
                .domainConfig(domainConfig)
                .build();
    }

    return new MiniConfigData
                .Builder()
                .build();
}
```

8. Customise the userAgent of the WebView in the mini program.

Customisation can be achieved by overriding the configData method of BaseMiniAppProxyImpl and intercepting the configType as MiniConfigData.TYPE_WEBVIEW.

Sample code:

```
@Override
public MiniConfigData configData(Context context, int configType, JSONObject params
    if(configType == MiniConfigData.TYPE_WEBVIEW) {
        //webView userAgent
        String ua = params.optString(MiniConfigData.WebViewConfig.WEBVIEW_CONFIG_UA
        MiniConfigData.WebViewConfig webViewConfig = new MiniConfigData.WebViewConf
        //Set the userAgent that the developer needs to append, note: don't set the
        webViewConfig.userAgent = "key/value";

        return new MiniConfigData
                .Builder()
                .webViewConfig(webViewConfig)
                .build();
    }

    return new MiniConfigData
            .Builder()
            .build();
}
```

9. Customised scanning capabilities

The mini program SDK provides a default scanning implementation (refer to Extension Library Support - Sweep), and also provides an interface for users to customise the code scanning ability.
To customise the scanning ability, you need to override the enterQRCode method of BaseMiniAppProxyImpl.

API description:

Parameter context: the current context of the mini program process that initiates the code scanning;
parameter onlyFromCamera: if or not only camera is allowed to sweep code;
parameter AsyncResult: the callback of code scanning result;
Return value boolean: true means custom code scanning ability, false means use built-in code scanning ability (need to integrate code scanning extension library).

```
/**
 * Scanning QR code
 *
 * @param context context
 * @param onlyFromCamera Only allow camera to scan code
 * @param result Scanning result
 * @return true:custom scanning;false:use built-in scanning
 */
@Override
public boolean enterQRCode(Context context, boolean onlyFromCamera, AsyncResult res
```

10. Specify the storage path where the mini program saves the file

By overriding the getSaveFileDir method of BaseMiniAppProxyImpl, we can implement the mini program API to save pictures and videos to the specified directory in the system album.

Sample code:

```
/**
 * Specify the directory where the mini program API saves images and videos to the
 *
 */
public  String getSaveFileDir(){
return "myapp";
}
```

Note：

The default path is tcmpp.

11. Listening to the mini program lifecycle

You can listen to the mini program lifecycle by overriding the onAppStateChange method of BaseMiniAppProxyImpl.

API description:

Parameter appState: life cycle state of the current mini program, refer to AppState;

parameter MiniAppEvent: life cycle event of the current mini program.

```
/**
 * Mini-program lifecycle event callbacks
 * Call context environment: main process UI threads
 *
 * @param appState event state
 * @param event event
 */
public abstract void onAppStateChange(@AppState int appState, MiniAppEvent event);
```

12. Customising the base library update policy and listening

You can override BaseMiniAppProxyImpl's isUpdateBaseLib method to listen for updates to the mini program's base library.

API description:

You can override BaseMiniAppProxyImpl's isUpdateBaseLib method to listen for updates to the mini program's base library.

```
/**
 * Whether or not to update when the base library detects an update
 * @param context
 * @param data Base library information data
 * @return true:update;false:don't update, default is true
 */
public abstract boolean isUpdateBaseLib(Context context, JSONObject data);
```

Custom Sharing Capabilities

Last updated：2025-01-16 19:11:04

The mini program SDK provides an API for sharing feature. Users can customize the sharing capability as follows:

1. Add a custom share item to the capsule control panel via MiniAppProxyImpl:

```
private static final String SHARE_TWITTER = "twitter";
/**
 * Returns a map of custom share data.
 * Call environment: sub-process
 *
 * key: Matches the MoreItem.id added in the getMoreItems method.
 * value: Matches the MoreItem.shareKey added in the getMoreItems method.
 * @
 */
@Override
public Map<String, Integer> getCustomShare(){
   Map<String, Integer> objects = new HashMap<>();
   objects.put(SHARE_TWITTER, ShareProxyImpl.OTHER_MORE_ITEM_2);
   return objects;
}

/**
 * Go back to the **More panels** button of the capsule. Set the ID of the extensio
 * Call environment: sub-process
 *
 * @param miniAppContext Mini program runtime environment (mini program process but
 * @param builder
 * @
 */
@Override
public ArrayList<MoreItem> getMoreItems(IMiniAppContext miniAppContext, MoreItemLis
 MoreItem item2 = new MoreItem();
 item2.id = ShareProxyImpl.OTHER_MORE_ITEM_2;
 item2.text = getString(miniAppContext,
                   R.string.applet_mini_proxy_impl_other2);
 item2.shareKey = SHARE_TWITTER;//Custom share key, must be set and unique, used wh
 item2.drawable = R.mipmap.mini_demo_about;

 builder.addMoreItem(item2)
 return builder.build();
}
```

2. SDK has built-in sharing buttons for QQ, Qzone, WeChat, and WeChat Circle of Friends, which can be quickly configured by overriding getDefaultShare.

```
 /**
 * Returns a list of built-in share buttons
 * Calling environment: child process
 *
 * @return Enumeration list of built-in share buttons
 */
@Override
public List<String> getDefaultShare() {
    ArrayList<String> defaultShare = new ArrayList<>();
    defaultShare.add(MoreItemList.SHARE_QQ);
    defaultShare.add(MoreItemList.SHARE_QZONE);
    defaultShare.add(MoreItemList.SHARE_WX_FRIENDS);
    defaultShare.add(MoreItemList.SHARE_WX_MOMENTS);
    return defaultShare;
}
```

3. To create a capsule, in theMore panel, clickListeners.

```
 /**
 * Return to Capsule More Panel Button Click Listener
 *
 * Return
 */
Override
public OnMoreItemSelectedListener getMoreItemSelectedListener() {
    return new DemoMoreItemSelectedListener(); }
}


public class DemoMoreItemSelectedListener extends DefaultMoreItemSelectedListener {
    public static final int CLOSE_MINI_APP = 150; @Override


    @Override
    public void onMoreItemSelected(IMiniAppContext miniAppContext, int moreItemId)
        //Handle developer-defined click events (except for custom sharing events)
        switch (moreItemId) {
            case CLOSE_MINI_APP: close(mini_APP)
                close(miniAppContext); return
                Returns;
            case OTHER_MORE_ITEM_1:
                miniAppContext.getAttachedActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(miniAppContext.getAttachedActivity(), "Custo
                    }
```

```
                });
                Return; }
        }


        // Handle built-in sharing and developer custom sharing, e.g. microblogging
        super.onMoreItemSelected(miniAppContext, moreItemId)}; return; }// Handle b
    }
}
```

4. Receive sharing according to the following types, click the event, the developer can get the sharing data in the share method, and call the third-party SDK to implement the sharing.

```
@ProxyService(proxy = ShareProxy.class)
public class ShareProxyImpl extends BaseShareProxy {
  /**
   * Share
   *
   * @param shareData shareData
   */
  @Override
  public void share(Activity activity, ShareData shareData) {
  //todo share
  }
}
```

Custom Authorization List

Last updated：2024-11-21 17:31:55

To get the authorization list, use the TmfMiniSDK as shown below:

```
/**
 * Get the mini program authorization list
 * @param appId
 * @param appVerType Mini program version type
 * @
 */
public static List<MiniAuthState> getAuthStateList(String appId, int appVerType)

/**
 * Set the authorization status
 * @param appId
 * @param appVerType Mini program version type
 * @param scopeName Permission name
 * @param grant Whether to authorize
 */
public static void setAuthState(String appId, int appVerType, String scopeName, boo
```

Custom Permission Requests

Last updated：2024-11-21 17:32:20

Mini program requests system permissions

When using mini programs, some APIs require not only mini program authorization but also corresponding Android system permissions to function correctly. By default, the SDK will automatically prompt the user to grant these system permissions when needed. However, you can disable this automatic permission request during SDK initialization. In this case, the host app must handle these system permission requests to ensure the mini program APIs work properly.

```
@ProxyService(proxy = MiniConfigProxy.class)
public class MiniConfigProxyImpl extends MiniConfigProxy {
    /**
     * App Application
     * @
     */
    @Override
    public Application getApp() {
        return "app Application";
     }

      /**
     * Create initialization configuration information
     * @
     */
    @Override
    public MiniInitConfig buildConfig() {
      MiniInitConfig.Builder builder = new MiniInitConfig.Builder();
      MiniInitConfig config = builder
                .configAssetName("tcmpp-android-configurations.json") //Configuratio
                 .imei("IMEI") //(Optional) Device ID, which is used for grayscale r
                 .autoRequestPermission(false) //onfigure whether to automatically r
                 .debug(true) //Log switch, turned off by default.
                 .build();
     }
}
```

Customize system permission requests

By implementing the IPermissionManagerProxy API, you can intercept and customize the logic for mini program system permission requests. The IPermissionManagerProxy API includes three methods: isPermissionGranted to check if the host has a specific system permission, and requestForPermission and requestForPermissions to notify the host to request one or multiple permissions.

Note：

Once you customize system permission requests, the autoRequestPermission setting in the initialization configuration will no longer be effective.

```
@ProxyService(proxy = IPermissionManagerProxy.class)
public class PermissionProxyImpl implements IPermissionManagerProxy {

    /**
     * Check if the host has a specific system permission.
     * @param context   Android context
     * @param permission    The system permission to check, refer to android.Manifes
     * @return              Whether the host has permission
     */
    @Override
    public boolean isPermissionGranted(Context context, String permission) {
        return ContextCompat.checkSelfPermission(context, permission) == PackageMana
     }

    /**
     * Notify the host to request a system permission.
     * @param activity      The mini program Activity requesting the permission
     * @param permission     The system permission to request, refer to android.Mani
     * @param callbacks      Callback to return the permission request result to the
     */
    @Override
    public void requestForPermission(Activity activity, String permission, RequestPe
        Toast.makeText(activity,"applying for" + permission + "permission", Toast.LE
     }

    /**
     * Notify the host to request multiple system permissions.
     * @param activity      The mini program Activity requesting the permissions
     * @param permissions    The list of system permissions to request, refer to and
     * @param callbacks     Callback to return the permission request result to the
     */
    @Override
    public void requestForPermissions(Activity activity, String[] permissions, Reque
        Toast.makeText(activity,"applying for" + permissions[0] + "permissions," Toa
     }
}
```

```
/**
 * Notify the result of system permission requests.
 */
interface RequestPermissionCallback {
    /**
```

```
     All system permissions are successfully granted.
     */
    void onSuccess();

    /**
     * Some or all system permissions are denied.
     * @param rejected      List of denied system permissions
     */
    void onFail(String[] rejected);
}
```

Custom User Attributes

Last updated：2024-11-21 17:33:22

Host app developers can set user attributes (such as region or account information) using methods provided by the SDK. This is useful for scenarios like targeted push notifications.

Setting user ID

API description: The userId parameter is used to set account information.

```
/**
 * Set the account information
 * @param userId
 */
 public static void setUserId(String userId)
```

Set location information

API description:

Parameter country indicates the user's country.

Parameter province indicates the user's province.

Parameter city indicates the user's city.

```
/**
 * Set the location information
 * @param country
 * @param province
 * @param city
 */
 public static void setLocation(String country, String province, String city)
```

Logging and Event Reporting

Last updated：2025-01-16 19:12:37

Custom log output implementation

By implementing the LogProxy API, you can control the internal logging of the mini program SDK.

Sample code:

```
@ProxyService(proxy = LogProxy.class)
public class LogProxyImpl extends LogProxy {

    @Override
    public void log(int logLevel, String tag, String msg, Throwable t) {
        switch (logLevel) {
            case Log.DEBUG:
                if (t == null) {
                    android.util.Log.d(tag, msg);
                } else {
                    android.util.Log.d(tag, msg, t);
                }
                break;
            case Log.INFO:
                if (t == null) {
                    android.util.Log.i(tag, msg);
                } else {
                    android.util.Log.i(tag, msg, t);
                }
                break;
            case Log.WARN:
                if (t == null) {
                    android.util.Log.w(tag, msg);
                } else {
                    android.util.Log.w(tag, msg, t);
                }
                break;
            case Log.ERROR:
                if (t == null) {
                    android.util.Log.e(tag, msg);
                } else {
                    android.util.Log.e(tag, msg, t);
                }
                break;
            default:
```

```
                if (t == null) {
                    android.util.Log.v(tag, msg);
                } else {
                    android.util.Log.v(tag, msg, t);
                }
                break;
        }
    }

    @Override
    public boolean isColorLevel() {
        return true;
    }
}
```

Custom Event Reporting

The host App can override the internal reporting logic of SDK by implementing the reportMiniAppEvent method of MiniAppProxy to customize the mini program event reporting.

Note：

Includes mini program operation events and the data reported by calling wx.reportEvent inside the mini program.

API is as follows:

```
/**
 * Agent for reporting mini program events.
 * @param eId eventId
 * @param eventName event name
 * @param props event property
 * @param app Mini program information.
 * @return is true, the SDK internal reporting logic is no longer executed, and the
 */
public abstract boolean reportMiniAppEvent(int eId, String eventName, JSONObject pr
```

The built-in event IDs are described below:

```
/** Custom event */
public static final int EID_None = 0;

/** Open a mini program */.
public static final int EID_OPEN_MINIAPP = 1;

/** Update mini program */ public static final int EID_OPEN_MINIAPP = 1.
public static final int EID_UPDATE_MINIAPP = 2;
```

```
/** Download mini program */
public static final int EID_DOWNLOAD_MINIAPP = 3;

/** Mini-program page view */
public static final int EID_MINIAPP_PAGE_VIEW = 4;

/** Exit mini program */ public static final int EID_MINIAPP_PAGE_VIEW = 4.
public static final int EID_EXIT_MINIAPP = 5;

/** Behavioural event of mini program */
public static final int EID_MINIAPP_ACTION = 6;
```

Customised Log Reporting

Real Time Log Reporting

The host APP can **customize the mini program real-time event log reporting logic by implementing the reportRealTimeLog method of MiniAppProxy.**

Note：

Include the log data written by the mini program internal call wx.getRealtimeLogManager.

API is as follows:

```
/**
 * Mini program real-time log upload agent
 * @param page Current page
 * @param jsLibVersion base library version
 * @param app mini program information
 * @param filterMsgs filter keywords
 * @param logs logs
 * @return is true, the SDK internal reporting logic is no longer executed.
 */
public abstract boolean reportRealTimeLog(String page, String jsLibVersion, MiniApp
                                          ArrayList<RealTimeLogItem> logs);
```

Internal log reporting for mini programs

The host APP can **customise the complaint feedback page of the mini program to collect the log upload logic by implementing the uploadUserLog method of MiniAppProxy.**

Note：

This includes log data written by the mini program's internal call to wx.getLogManager, and the user can upload printed logs by using the button component's open-type="feedback".

```
/**
```

```
  * feedback log upload
  *
  * @param appId mini program appId
  * @param logPath The log path.
  * @return is true, the SDK internal upload logic is not executed.
  */
public abstract boolean uploadUserLog(String appId, String logPath);
```

Open APIs

Last updated：2025-04-03 17:53:12

The mini program SDK provides some open APIs for implementing features that are provided by host apps, such as login, user information retrieval, and payment.

Note:

Starting from version 2.1.0, the SDK APIs come with default implementations for login (wx.login), user information (wx.getPhoneNumber, wx.getEmail, wx.chooseAvatar, wx.getNickName), and session status checks (wx.checkSession). This allows developers to easily configure and integrate user login and host application user information logic for mini programs.
Default implementation since version 2.1.0 is supported only on SaaS.

SDK version 2.1.0 and later

The custom open APIs supported by SDK v2.1.0 and later versions are shown in the table below:

Mini program method	MiniOpenApiProxyV2 implementation method	Description	Default implementation
wx.login	login	Login API SDK	Default implementation from version 2.1.0
wx.getUserInfo	getUserInfo	Retrieves basic user information	No default implementation
wx.getUserProfile	getUserProfile	Retrieves user profile information	No default implementation
wx.getPhoneNumber	getPhoneNumber	Retrieves phone number	Default implementation from version 2.1.0
wx.requestPayment	requestPayment	Initiates a payment	No default implementation
wx.requestMidasPaymentGameItem	requestMidasPaymentGameItem	request mini	No default

		game payment	implementation from version 2.2.4
wx.checkSession	checkSession	Checks if the login session is expired	Default implementation from version 2.1.0
wx.getEmail	getEmail	Retrieves user email	Default implementation from version 2.1.0
wx.chooseAvatar	chooseAvatar	Retrieves the user profile photo	Default implementation from version 2.1.0
wx.getNickName	getNickName	Retrieves the user nickname	Default implementation from version 2.1.0

App developers can override the MiniOpenApiProxyV2 proxy methods to implement logic for retrieving basic user information (wx.getUserInfo), user profile information (wx.getUserProfile), and initiating payments (wx.requestPayment) ,and mini game payments (wx.requestMidasPaymentGameItem).

```
@ProxyService(proxy = MiniOpenApiProxyV2.class)
public class MiniOpenApiProxyImplV2 extends MiniOpenApiProxyDefault {
    private static final String TAG = "MiniOpenApiProxyImplV2";

    @Override
    public void getUserInfo(IMiniAppContext miniAppContext, JSONObject params, Asyn
        QMLog.d(TAG, "getUserInfo:" + params);
        JSONObject jsonObject = new JSONObject();
        try {
            final JSONObject userInfo = new JSONObject();
            TmfMiniSDK.callMainProcessPlugin(OpenDataIPC.OPEN_DATA_IPC_EVENT_GET_US
                @Override
                public void result(boolean b, Bundle bundle) {
                    try {
                        userInfo.put("nickName", bundle.getString("userId"));
                        userInfo.put("avatarUrl", bundle.getString("avatarUrl"));
                    } catch (JSONException e) {
```

```
                    e.printStackTrace();
                }
            }
        });
        userInfo.put("gender", 0);
        userInfo.put("country", "CN");
        userInfo.put("province", "BeiJing");
        userInfo.put("city", "BeiJing");
        userInfo.put("language", "en");
        jsonObject.put("userInfo", userInfo);
    } catch (JSONException e) {
        e.printStackTrace();
    }
    result.onReceiveResult(true, jsonObject);
}

@Override
public void getUserProfile(IMiniAppContext miniAppContext, JSONObject params, A
    QMLog.d(TAG, "getUserProfile:" + params);
    JSONObject jsonObject = new JSONObject();
    try {
        final JSONObject userInfo = new JSONObject();
        TmfMiniSDK.callMainProcessPlugin(OpenDataIPC.OPEN_DATA_IPC_EVENT_GET_US
            @Override
            public void result(boolean b, Bundle bundle) {
                try {
                    userInfo.put("nickName", bundle.getString("userId"));
                    userInfo.put("avatarUrl", bundle.getString("avatarUrl"));
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        });
        userInfo.put("gender", 0);
        userInfo.put("country", "CN");
        userInfo.put("province", "BeiJing");
        userInfo.put("city", "BeiJing");
        userInfo.put("language", "en");
        jsonObject.put("userInfo", userInfo);
    } catch (JSONException e) {
        e.printStackTrace();
    }
    result.onReceiveResult(true, jsonObject);
}

@Override
public void requestPayment(IMiniAppContext miniAppContext, JSONObject params, A
```

```
        QMLog.d(TAG, "requestPayment:" + params);
        JSONObject jsonObject = new JSONObject();
        try {
            jsonObject.put("key", "wx.requestPayment");
        } catch (JSONException e) {
            e.printStackTrace();
        }
        PaymentManager.g(miniAppContext.getContext()).startPayment(miniAppContext,
    }

    @Override
    public void requestMidasPaymentGameItem(IMiniAppContext miniAppContext, JSONObj
        // call your custom payment implementation
        boolean paySuccess = PaymentManagerV2.g().startMidasPayment(miniAppContext,
        // notify payment result with AsynResult
        if(paySuccess){
            result.onReceiveResult(true,successData);
        }else{
            result.onReceiveResult(false,failedData);
        }
    }
}
```

To use the SDK's default login logic, developers need to override the getAccount method of MiniAppProxy to pass the host application's UID to the SDK. The SDK will use this UID to exchange and retrieve user information from the mini program's backend service.

```
@ProxyService(proxy = MiniAppProxy.class)
public class MiniAppProxyImpl extends BaseMiniAppProxyImpl {
        @Override
          public String getAccount() {
                        // User identifier of the host application
                    String uid = getHostAppUserId();
                    return uid;
      }
}
```

SDK version 2.0.15 and earlier

For SDK version 2.0.15 and earlier, the following table shows the custom open APIs:

Mini program method	MiniOpenApiProxy method	Description
wx.login	login	Login API

wx.getUserInfo	getUserInfo	Retrieves basic user information
wx.getUserProfile	getUserProfile	Retrieves user profile information
wx.getPhoneNumber	getPhoneNumber	Retrieves phone number
wx.requestPayment	requestPayment	Initiates a payment
wx.checkSession	checkSession	Checks if the login session is expired

Users can implement the MiniOpenApiProxy proxy to link data interactions between the mini program and the host application. See the example below:

Notes:

IMiniAppContext provides the current context information of the mini program.

JSONObject contains the parameters passed when the mini program calls the open APIs.

AsyncResult is an asynchronous callback API used to return the host application's open API results back to the mini program.

```
@ProxyService(proxy = MiniOpenApiProxy.class) public class MiniOpenApiProxyImpl imp
```

Compatibility of custom login logic

Since SDK version 2.1.0 and later come with built-in login and user information retrieval logic, existing customers who have already implemented custom user login logic by overriding the MiniOpenApiProxy proxy need to configure the SDK to continue using their custom logic after updating to version 2.1.0 or later. Here’s how to do it:

```
@ProxyService(proxy = MiniOpenApiProxyV2.class)
public class DisableDefaultOpenApiProxyImpl extends MiniOpenApiProxyV2 {
    @Override
    public boolean disableV2API(IMiniAppContext miniAppContext) {
        // Return true to disable the default login implementation, false to use th
        // Default value: false.
        return true;
    }
}
```

Others

Last updated：2024-11-21 17:34:28

Process special URLs in web-view component

If the webpage displayed in the web-view component contains special URLs, such as those starting with a custom scheme like tcmpp://host/path, the host app can intercept and customize the redirections of the URLs.
To intercept URL navigation, create a class that implements the IWebViewLinkProxy API and add the ProxyService annotation:

```
@ProxyService(proxy = IWebViewLinkProxy.class)
public class CustomWebViewLinkProxy implements IWebViewLinkProxy {

    /**
     * Handle custom url loading in web-view component.
     * Custom url is url with any custom scheme (scheme not network protocol such a
     * @param miniContext context of mini-program
     * @param url the url which is loading
     * @return return true if the url loading is handled, otherwise false the web-v
     */
    @Override
    public boolean webViewCustomUrlLoading(IMiniAppContext miniContext, String url)
        Uri uri = Uri.parse(url);
        if ("tcmpp".equals(uri.getScheme())) {
            Intent intent = new Intent(Intent.ACTION_VIEW, uri);
            miniContext.getAttachedActivity().startActivity(intent);
            return true;
        }
        return false;
    }
}
```

In the implementation, the mini program context and the loaded URL are provided. If the developer handles the URL redirections and returns true, the web-view component will no longer process the URL. If false is returned, the web-view component will process the URL according to normal logic.

API Description

Last updated：2025-06-06 18:27:19

MiniStartOptions

Configuration options for launching a mini program

```
/**
 * Whether to force update when opening the mini program (effective only on the fir
 */
public boolean isForceUpdate = false;
/**
 * Entry address, supports adding parameters: path?key=value&key1=value1
 */
public String entryPath;
/**
 * Receives error messages during the mini program launch process
 */
public ResultReceiver resultReceiver;

/**
 * Mini program launch parameters
 */
public String params;

/**
 * Sets the task mode to be used when opening the mini program
 *
 * false: Multi-task mode; true: Single-task mode
 */
public boolean isSingleTask;
```

MiniCode

Error code descriptions.

```
/**
 * Success
 */
public static final int CODE_OK = 0;
```

```
////////////////////////////// Server error ////////////////////////////////
public static final int C_SERVER = -11000;
/**
 * Shark network error
 */
public static final int C_SERVER_SHARK_ERROR = -11001;
/**
 * Server returned an error code
 */
public static final int C_SERVER_RET_CODE_ERROR = -11002;
/**
 * Server response is null
 */
public static final int C_SERVER_RESPONSE_NULL = -11003;
/**
 * Server returned an invalid mini program update type
 */
public static final int C_SERVER_UPDATE_TYPE_ERROR = -11004;
/**
 * Server data parsing exception
 */
public static final int C_SERVER_PARSE_DATA_ERROR = -11005;
/**
 * The mini program does not exist or has been removed
 */
public static final int C_SERVER_TAKE_OFF = -11006;
/**
 * Monthly active user limit reached
 */
public static final int C_SERVER_MAU_LIMIT = -11007;
/**
 * Resource limit reached
 */
public static final int C_SERVER_RES_LIMIT = -11008;
/**
 * Single API request frequency too high
 */
public static final int C_SERVER_FREQ_LIMIT_API = -11011;
/**
 * Total request frequency too high
 */
public static final int C_SERVER_FREQ_LIMIT_TOTAL = -11012;

////////////////////////////// Client error ////////////////////////////////
public static final int C_CLIENT = -12000;
/**
 * The Shark instance is null
```

```
 */
public static final int C_CLIENT_SHARK_IS_NULL = -12001;
/**
 * Previewing the mini program requires login first
 */
public static final int C_CLIENT_NEED_LOGIN_PREVIEW_APP = -12002;
/**
 * Data parsing exception
 */
public static final int C_CLIENT_JSON_EXCEPTION = -12003;
/**
 * Mini program information is missing
 */
public static final int C_CLIENT_MINI_APP_INFO_ERROR = -12005;
/**
 * QR code error
 */
public static final int C_CLIENT_QRCODE_ERROR = -12006;
/**
 * Invalid mini program QR code
 */
public static final int C_CLIENT_QRCODE_INVALIDATE = -12007;
/**
 * The appid is empty
 */
public static final int C_CLIENT_APPID_EMPTY = -12008;
/**
 * Business ID is null
 */
public static final int C_CLIENT_QRCODE_BUSINESSID_NULL = -12009;
/**
 * Mini program launch exception
 */
public static final int C_CLIENT_START_MINI_APP_THROWABLE = -12010;
/**
 * JSON parsing error
 */
public static final int C_CLIENT_JSON_ERROR = -12011;
/**
 * Mini program download failed
 */
public static final int C_CLIENT_MINI_APP_DOWNLOAD_FAIL = -12012;
/**
 * Mini program parsing failed
 */
public static final int C_CLIENT_MINI_APP_PARSE_FAIL = -12013;
/**
```

```
 * MBEngine not integrated, game not supported
 */
public static final int C_CLIENT_MINI_GAME_MBENGINE_LOAD_FAIL = -12100;
/**
 * Mini game disabled by server
 */
public static final int C_CLIENT_MINI_GAME_DISABLED = -12101;
/**
 * Mini game base library load failed
 */
public static final int C_CLIENT_MINI_APP_BASE_LIB_LOAD_FAIL = -12103;
/**
 * The game.js or the specified entry file does not exist
 */
public static final int C_CLIENT_MINI_GAME_ENTRY_ERROR = -12104;
```

MiniApp

Mini program information.

```
/**
 * Mini program version: Released
 */
public static final int TYPE_ONLINE = MiniSDKConst.ONLINE;
/**
 * Mini program version: Development
 */
public static final int TYPE_DEVELOP = MiniSDKConst.DEVELOP;
/**
 * Mini program version: Preview
 */
public static final int TYPE_PREVIEW = MiniSDKConst.PREVIEW;
/**
 * Mini program appid
 */
public String appId;
/**
 * Mini program version (released/preview/development)
 */
public int appVerType;
/**
 * Mini program version
 */
```

```
public String version;
/**
 * Mini program name
 */
public String name;
/**
 * Mini program icon
 */
public String iconUrl;
/**
 * Mini program introduction
 */
public String appIntro;
/**
 * Developer's company name
 */
public String appDeveloper;
/**
 * Timestamp
 */
public long time;
/**
 * Mini program engine type:
 * MiniEngineType.MiniApp: Mini program
 * MiniEngineType.MiniGame:  Mini game
 */
public MiniEngineType engineType;
```

MiniScene

The scene in which the mini program is opened.

```
/**
 * Main entry point of the mini program, from the "Recently used" list
 */
public static final int LAUNCH_SCENE_MAIN_ENTRY = 1001;
/**
 * Opened via QR code scan
 */
public static final int LAUNCH_SCENE_QR_CODE_FROM_SCAN = 1011;
/**
 * Opened via search
 */
public static final int LAUNCH_SCENE_SEARCH = 2005;
```

SearchOptions

```
/**
 * Search keyword; if empty, searches all mini programs
 */
public String keyWord = "";
/**
 * Specifies the first-level category
 */
public String firstLevelCate;
/**
 * Specifies the second-level category
 */
public String secondaryLevelCate;
/**
 * Specifies the engine type:
 * 1. MiniEngineType.MiniApp:  Searches mini programs only
 * 2. MiniEngineType.MiniGame:  Searches mini games only
 * 3. If not specified, searches both mini programs and mini games
 */
public MiniEngineType engineType;
```

ShareData

```
/**
     * Share source, value from ShareSource
     */
    public int shareSource;
    /**
     * Share target, value from ShareTarget
     */
    public int shareTarget;
    /**
     * ID set in the share panel, used to distinguish sharing channels
     */
    public int shareItemId;
    /**
     * Share title
     */
    public String title;
    /**
     * Share summary
```

```
     */
    public String summary;
    /**
     * TPath of the share image.  Can be a local image path or a network image path
     */
    public String sharePicPath;
    /**
     * Whether it is a local image.  If true, sharePicPath is the local image path;
     */
    public boolean isLocalPic;
    /**
     * Field obtained from the server: Share link
     */
    public String targetUrl;
    /**
     * Mini program package information
     */
    protected MiniAppInfo miniAppInfo;
```

ShareSource

```
public static class ShareSource {

    public static final int INNER_BUTTON = 11; // From an internal button of the mi
    public static final int MORE_BUTTON = 12; // From the More option in the capsul
}
```

ShareTarget

```
public static class ShareTarget {


    public static final int WECHAT_FRIEND = 3; // Shares to WeChat contact
    public static final int WECHAT_MOMENTS = 4; // Shares on WeChat Moments
}
```

ShareResult

```
public static class ShareResult {
    public static final int SUCCESS = 0;// Share successful
    public static final int FAIL = 1;// Share failed
    public static final int CANCEL = 2;// Share cancelled
}
```

MiniStartLinkOptions

```
public class MiniStartLinkOptions {
    /**
     * Whether to force update when opening the mini program (effective only on the
     */
    public boolean isForceUpdate = false;
    /**
     * Entry path
     */
    public String entryPath;
    /**
     * Receives error messages during the mini program launch process
     */
    public ResultReceiver resultReceiver;

    /**
     * Mini program launch parameters
     */
    public String params;
}
```

MiniInitConfig

```
/**
 * Name of the configuration file in the assets
 */
private String configAssetName;
/**
 * Custom configuration file path
 */
private String configFilePath;
/**
 * Configuration file content
```

```
 */
private String configJsonStr;
/**
 * SDK log switch
 */
private boolean debug;
/**
 * Configures an external shark instance
 */
private IShark shark;
/**
 * Whether to verify the package name in the configuration file when loading it
 */
private boolean verifyPkg;
/**
 * Whether to use Tencent Browsing Service X5 kernel
 */
private boolean isUserX5Core = true;
/**
 * Whether to force the use of the core base library from assets
 */
private boolean forceUseBaseLibInAsset;
/**
 * Path for preset offline package assets
 */
private String assetPathOfPresets;
```

IMiniAppContext

```
/**
 * Returns mini program information
 */
MiniAppInfo getMiniAppInfo();
```

MiniAppInfo

```
public String appId; // Mini program appid
public String name; // Mini program name
public String iconUrl; // URL of the mini program icon
public String version; // Mini program version number
```

```
public int verType; // Mini program type: development, preview, or released version
```

IpcCallback

```
public interface IpcCallback {
    /**
     * Callback for process communication
     * @param isSucc - Whether the callback was successful
     * @param response - The return data
     */
    void result(boolean isSucc, Bundle response);
}
```

IpcRequestEvent

```
public Context context;
// Data associated with the request
public Bundle data;
// The callback for the response
public IpcCallback callback;
```

RequestEvent

```
// Reference to the mini program activity
public WeakReference<Activity> activityRef;
// Event name
public String event;
// Event parameters
public String jsonParams;
```

AppState

```
/**
 * Mini program starting
```

```
 */
int STATE_START = 1;
/**
 * Mini program switched to the foreground
 */
int STATE_FOREGROUND = 2;
/**
 * Mini program closed using the capsule button
 */
int STATE_CLOSE = 3;
/**
 * Mini Program switched to the foreground
 */
int STATE_BACKGROUND = 4;
/**
 * Mini program destroyed
 */
int STATE_DESTROY = 5;
```

MiniAppEvent

```
/**
 * Mini program information
 */
public MiniApp miniApp;
/**
 * Whether it is a hot start
 */
public boolean isHotStart;
```

PreDownloadInfo

```
/**
 * Mini program appid
 */
public String appId;
/**
 * Whether to download the mini program package
 */
public boolean isDownload;
```

IDownloadCallback

```
/**
 * Callback for successful download
 *
 * @param downloadInfo
 */
void onFinish(DownloadInfo downloadInfo);

/**
 * Callback for failed download
 *
 * @param downloadInfo
 */
void onError(DownloadInfo downloadInfo);
```

DownloadInfo

```
// IOEXCEPTION and IllegalAccessException occur during the download process
public static final int CODE_DOWNLOAD_IOEXCEPTION = -100001;
// An exception occurs during the download process
public static final int CODE_DOWNLOAD_EXCEPTION = -100002;
// No network connection
public static final int CODE_NO_NETWORK = -100003;
// Invalid download parameters
public static final int CODE_PARAM_ERROR = -100004;
// Download directory creation failed
public static final int CODE_DOWNLOAD_DIR_CREATE_FAIL = -100005;
// Mini program package parsing failed
public static final int CODE_MINI_APP_PARSE_FAIL = -12013;

/**
 * Mini program appid
 */
private String appId;
/**
 * Error code
 */
private int errCode;
/**
 * Error message
 */
```

```
private String message;
```

IMiniAppFileManager

```
/**
 * Gets the absolute path of a wx file
 * @param wxFilePath
 * @return
 */
String getAbsolutePath(String wxFilePath);

/**
 * Converts an absolute path to a wx file
 * @param path
 * @return
 */
String getWxFilePath(String path);

/**
 * Gets a temporary directory
 * @param suffix
 * @return
 */
String getTmpPath(String suffix);
```

BaseJsPlugin

```
/**
 * Mini program context
 */
protected IMiniAppContext mMiniAppContext;
/**
 * Mini program information
 */
protected MiniAppInfo mMiniAppInfo;
/**
 * Mini program package information
 */
protected ApkgInfo mApkgInfo;
```

MiniEngineType

```
public enum MiniEngineType {
    MiniApp, // Mini program
    MiniGame // Mini game
}
```

ChooseMediaProxy.IChooseMediaListener

```
interface IChooseMediaListener {
    /**
     * @param result Whether the media file is successfully selected, true indicate
     * @param uris List of selected media files, returns empty list if selection fa
     */
    void onResult(boolean result, ArrayList<Uri> uris);
}
```

ChooseMediaOptions

```
public class ChooseMediaOptions {
    public static final int SIZE_TYPE_ORIGINAL = 0x01; // Full-size image
    public static final int SIZE_TYPE_COMPRESS = 0x02; // Compressed image

    public static final int MEDIA_TYPE_IMAGE = 0; // Image
    public static final int MEDIA_TYPE_VIDEO = 1; // Video
    public static final int MEDIA_TYPE_MIX = 2; // Mixed mode

    public static final int CAMERA_TYPE_FRONT = 0; // Front camera
    public static final int CAMERA_TYPE_BACK = 1; // Rear camera

    public int maxCount = 1; // Maximum number of files to select in multi-file sel
    public int sizeType = SIZE_TYPE_ORIGINAL; // Full-size or compressed, can use S
    public int mediaType = MEDIA_TYPE_IMAGE; // Image, video, or mixed mode
    public int duration; // Maximum video recording duration in seconds. The time r
    public int camera; // Use front or rear camera (only effective for chooseMediaF
    public Uri cameraOutput; // Output file path for camera mode. Non-null means th
}
```

SDK Extension Components

Last updated：2025-06-06 18:27:18

A mini program's main SDK (mini_core) provides the basic runtime capabilities of the mini program. In addition to mini_core, the mini program SDK also offers the following extension components, which can be integrated to the superapp as needed.

QR code scanning extension SDK

Component description: If your mini program uses the QR code scanning capability, you need to add the following SDK to support the scanning feature.
Integration method: Add the QR code scanning extension library dependency as follows:

```
// QR code scanning extension component
implementation 'com.tencent.tcmpp.android:mini_extra_qrcode:${version}' // For vers
```

After the QR code scanning extension SDK is added, the supported mini program APIs are as follows:

API name	Description
wx.scanCode	Opens the client’s QR code scanning interface

Required permissions:

Permission	Description
Camera	Camera permission is required for QR code scanning.
File read/write	File read/write permissions are required to recognize QR codes in local images.

Tencent Maps extension SDK

Component description: Developed for apps in the Chinese mainland. If your mini program uses the map capabilities, you need to add the following SDK.
Integration method: Add the map extension library dependencies as follows.

```
implementation 'com.tencent.tcmpp.android:mini_extra_map:${version}'// For version
Implementation 'com.tencent.map:tencent-map-vector-SDK:4.5.10' // For version infor
implementation 'com.tencent.map:sdk-utilities:1.0.7'
```

```
implementation 'com.tencent.map.geolocation:TencentLocationSdk-openplatform:7.4.7'
```

You need to configure the project in the Tencent Location Service console and obtain the API key required to access the Tencent Maps service. For detailed steps, see Development Guide.

Once you've completed the above steps, you need to configure your API key in the Android project. Add the following meta-data in the AndroidManifest.xml file, and replace (YOUR_API_KEY) with your actual API key:

```
<application
    ...
    <meta-data
    android:name="TencentMapSDK"
    android:value="(YOUR_API_KEY)" />
    ...
</application>
```

After the Tencent Maps extension SDK is added, the supported mini program APIs are as follows:

API name	Description
Map	Supports map APIs, including map display, location selection, and POI query, etc.

Required permissions:

Permission	Description
Location permission	Required to use location services for displaying map locations.

Google Maps and Huawei Maps extension SDKs

Component description: Developed for apps outside the Chinese mainland. If your mini app uses map capabilities, you need to add the following SDK.

Integration method: Add the map extension library dependencies as follows.

```
implementation 'com.tencent.tcmpp.android:mini_extra_google_map:${version}'// For v
implementation 'com.google.android.gms:play-services-maps:18.1.0'          // For v
implementation 'com.google.maps.android:android-maps-utils:2.3.0'
```

Since some Huawei devices do not support embedded Google Maps, the map may not be displayed. You can integrate Petal Map as a supplementary solution, and the mini program framework will prioritize the use of Petal Map on Huawei devices.

```
repositories {
```

```
  maven {url 'https://developer.huawei.com/repo/'}
}
implementation 'com.tencent.tcmpp.android:mini_extra_huawei_map:${version}'// For v
implementation 'com.huawei.hms:maps:6.9.0.300'                              // For v
implementation 'com.huawei.hms:maps-basic:6.9.0.300'
implementation 'com.huawei.hms:site:6.5.1.300'
```

For Google Maps, you need to configure the Google Cloud project in the Google Cloud console and obtain the API key required to access the Google Maps service. For detailed steps, see **Setup in the Cloud Console** and **Use API Keys**.

Once you've completed the above steps, you need to configure your API key in the Android project. Add the following meta-data in the AndroidManifest.xml file, and replace (YOUR_API_KEY) with your actual API key:

```
<application
    ...
    <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="(YOUR_API_KEY)" />
     ...
</application>
```

For Petal Map, you need to create a project in the Huawei Maps console, activate the map and location services, and obtain the API key used to access the location service. For detailed steps, see Configuring App Information in AppGallery Connect. Then follow the guide of Integrating the HMS Core SDK to download the “agconnect-services.json” file to your project and configure the Huawei AGC plugin.

To use Huawei's location service properly, you need to add the following meta-data to the AndroidManifest.xml file and replace (YOUR_API_KEY) with your actual key:

```
<application
    ...
    <meta-data
    android:name="HuaweiApiKey"
    android:value="(YOUR_API_KEY)" />
    ...
</application>
```

Note:

For security purpose, it is recommended to request a separate API key for location services.

After the Google Maps and Huawei Maps extension SDKs are added, the supported mini program APIs are as follows:

API name	Description
Map	Supports map APIs and components, including map display, location selection, and POI query, etc.

Required permissions:

Permission	Description
Location permission	Required to use location services for displaying map locations.

Live streaming component extension SDK

Component description: If you need to use the live streaming components (live-player and live-pusher) for developing capabilities of live streaming push and pull, you need to add the following SDKs.

Integration method: Add the live streaming component dependencies as follows.

```
// Live streaming component support library
implementation 'com.tencent.tcmpp.android:mini_extra_trtc_live:${version}'// For ve
// Live streaming component library
implementation 'com.tencent.liteav:LiteAVSDK_Professional:latest.release' // For ve
```

In addition to adding the above dependencies, you need to override the following methods of implementing baseminiAppProxyImpl and provide the licenseURL and licenseKey required for the live streaming component. This is necessary to complete the initialization configuration for the live streaming component. If you do not configure the correct LicenseUrl and LicenseKey, the live streaming component will not function properly.

Note:

For the method of obtaining the LicenseURL and LicenseKEY, see Adding and Renewing A License.

```
@ProxyService(proxy = MiniAppProxy.class)
public class MiniAppProxyImpl extends BaseMiniAppProxyImpl {
      @Override
    public MiniConfigData configData(Context context, int configType, JSONObject pa
        if(configType == MiniConfigData.TYPE_LIVE) {
            // Live streaming configuration
            MiniConfigData.LiveConfig liveConfig = new MiniConfigData.LiveConfig();
            // The following key and URL can only be used for demo
            liveConfig.licenseKey = "";
            liveConfig.licenseUrl = "";

            return new MiniConfigData
                    .Builder()
                    .liveConfig(liveConfig)
                    .build();
        }

        return null;
    }
}
```

After the live streaming extension SDK is added, the supported mini program APIs are as follows:

API name	Description
wx.createLivePusherContext	Creates the live streaming pusher context.
LivePusherContext	Supports LivePusherContext APIs.
wx.createLivePlayerContext	Creates a live streaming player context.
LivePlayerContext	Supports LivePlayerContext APIs.
live-pusher	Tag for live streaming push.
live-player	Tag for live streaming play.

Required permissions:

Permission	Description
Camera	-
Recording	-

LBS extension SDK

Component description: The LBS component provides capabilities related to location information, compass, accelerometer, positioning, and device motion.
Integration method: Add the LBS extension library dependency as follows.

```
implementation 'com.tencent.tcmpp.android:mini_extra_lbs:${version}'// For version
```

After the LBS extension SDK is added, the supported mini program APIs are as follows:

API name	Description
Location information	Supports location information APIs.
Compass	Supports compass APIs.
Accelerometer	Supports accelerometer APIs.
Device motion	Supports device motion APIs.
Gyroscope	Supports gyroscope APIs.

Required permissions:

Permission	Description
Location permission	Required to obtain the location.

Bluetooth extension SDK

Component description: After the Bluetooth extension library is added, you can use Bluetooth APIs.

Integration method: Add the Bluetooth extension library dependency as follows:

```
implementation 'com.tencent.tcmpp.android:mini_extra_bluetooth:${version}'// For ve
```

After the LBS extension SDK is added, the supported mini program APIs are as follows:

API	Description
Bluetooth - general-purpose	General-purpose Bluetooth API.
Bluetooth - Low Energy (BLE) peripheral device	Bluetooth Low Energy (BLE) peripheral devices APIs.
Bluetooth-Low Energy (BLE) central device	Bluetooth Low Energy (BLE) central devices APIs.
Bluetooth-beacon	Bluetooth beacon APIs.

Required permissions:

Permission	Description
Bluetooth	Required to operate the Bluetooth.
Location permission	Required to search the Bluetooth device.

NFC extension SDK

Component description: The NFC extension SDK can be added to enable NFC read/write capabilities.

Integration method: Add the NFC extension SDK dependency as follows.

```
implementation 'com.tencent.tcmpp.android:mini_extra_nfc:${version}'// For version
```

After the NFC extension SDK is added, the supported mini program APIs are as follows:

API name	Description

wx.getNFCAdapter	Gets the NFC operation management instance object.
NFCAdapter	Supports NFCAdapter APIs.
NFC instances (NFCA, NFCB, NFCV, NFCF, NDEF, IsoDep, MifareUltralight, MifareClassic)	Supports NFC tag instance APIs.

Required permissions:

Permission	Description
NFC permission	Required to access NFC capabilities.

Biometric authentication extension SDK

Component description: Biometric authentication extension SDK provides capabilities related to device fingerprinting and face recognition.

Integration method: Add the biometric authentication extension library dependency as follows.

```
implementation 'com.tencent.tcmpp.android:mini_extra_soter:${version}'// For versio
```

After the biometric authentication extension SDK is added, the supported mini program APIs are as follows:

API name	Description
wx.startSoterAuthentication	-
wx.checkIsSupportSoterAuthentication	-
wx.checkIsSoterEnrolledInDevice	-

Required permissions:

Permission	Description
Fingerprint access	Required to access fingerprint authentication.

Clipboard extension SDK

Component description: Provides clipboard access capability.

Integration method: Add the extension library dependency as follows.

```
implementation 'com.tencent.tcmpp.android:mini_extra_clipboard:${version}'// For ve
```

After the LBS extension SDK is added, the supported mini program APIs are as follows:

API name	Description
wx.getClipboardData	-
wx.setClipboardData	-

Required permissions:

Permission	Description
Clipboard permission	Required to access and modify clipboard data.

Contacts extension SDK

Component description: Provides capabilities for accessing contacts.

Integration method: Add the extension library dependency as follows.

```
implementation 'com.tencent.tcmpp.android:mini_extra_contact:${version}'// For vers
```

After the contacts extension SDK is added, the supported mini program APIs are as follows:

API name	Description
wx.addPhoneContact	Adds a contact.
wx.chooseContact	Selects a contact.

Required permissions:

Permission	Description
Contacts read/write permission	Required to access and write contacts information.

PDF extension SDK

Component description: Provides the capability to open PDF documents.

Integration method: Add the extension library dependency as follows.

```
implementation 'com.tencent.tcmpp.android:mini_extra_pdf:${version}'// For version
```

After the PDF extension SDK is added, the supported mini program APIs are as follows:

API name	Description
wx.openDocument	Opens the document (PDF format only).

Media extension SDK

Component description: Provides default implementations for chooseMedia and previewMedia. Starting from version 1.5.9, a lite version has been added that uses the system image picker to eliminate the need for READ_MEDIA_IMAGES and READ_MEDIA_VIDEO permissions. The lite version and the non-lite version are mutually exclusive, and integrating both will result in compilation errors. Please choose the appropriate version based on your needs.

Integration method for non-lite version:

```
implementation 'com.tencent.tcmpp.android:mini_extra_media_support:${version}'// Fo
```

Integration method for lite version:

```
implementation 'com.tencent.tcmpp.android:mini_extra_media_support_lite:${version}'
```

Implement the MediaImageLoaderProxy, use a custom image loading implementation for image loading in the mini_extra_media_support library.

Note:

You can also implement the MediaChooseJsProxy to customize the logic for chooseMedia.

```
@ProxyService(proxy = MediaImageLoaderProxy.class)
public class CustomMediaImageLoaderProxy implements MediaImageLoaderProxy {
    private GlideImageEngine glideImageEngine = new GlideImageEngine();

    @Override
    public ImageEngine getCustomImageEngine() {
        return glideImageEngine;
    }

    static class GlideImageEngine implements ImageEngine {

        @Override
        public void loadPhoto(@NonNull Context context, @NonNull Uri uri, @NonNull
            Glide.with(context).load(uri).transition(withCrossFade()).into(imageVie

        }
```

```
        @Override
        public void loadGifAsBitmap(@NonNull Context context, @NonNull Uri gifUri,
            Glide.with(context).asBitmap().load(gifUri).into(imageView);

        }

        @Override
        public void loadGif(@NonNull Context context, @NonNull Uri gifUri, @NonNull
            Glide.with(context).asGif().load(gifUri).transition(withCrossFade()).in
```

Mini game extension SDK

Component description: Provides the implementation for the mini game engine.

Integration method: Add the extension library dependency as follows.

```
implementation 'com.tencent.tcmpp.android:mini_extra_mbengine:${version}'// For ver
```

After the extension library is added, mini games are supported. For mini game APIs, see API Overview.

Google ads extension SDK

Component description: Provides Google AdMob ad loading capability.

Integration method: Add the extension library dependency as follows.

```
implementation 'com.tencent.tcmpp.android:mini_extra_admob:${version}'// For versio
```

Note:

This feature depends on AdMob. Please integrate it according to the official documentation first. For more information, see Get Started.

The mini program framework operates on a multi-task, multi-process architecture, with mini programs running in independent processes. Since AdMob does not support multi-process by default, the ad extension library has been adapted by modifying the declaration of AdMob's AdActivity. The details are as follows:

```
<activity
    android:name="com.google.android.gms.ads.AdActivity"
    android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|
    android:exported="false"
    android:taskAffinity=":admob"
    android:excludeFromRecents="true"
```

```
    android:multiprocess="true"
    android:theme="@android:style/Theme.Translucent"
    tools:ignore="MissingClass" />
```

Modified field	Purpose and impact	Removable?
android:multiprocess="true"	**Purpose**: Allows AdActivity to run in the calling process, enabling the mini program process to use AdMob ads. **Impact:** This attribute only affects the child process calling AdMob ads; it does not affect the main process.	No, removing it will prevent the mini programs and mini games from displaying ads properly.
android:taskAffinity=":admob"	**Purpose**: AdMob adds the FLAG_ACTIVITY_NEW_TASK tag when clicking banner ads, causing AdActivity to be pushed into the superapp's main task stack instead of the mini program's task stack. As a result, when returning from the target page, it will navigate back to the superapp's main stack instead of the mini program page. Adding this attribute allows AdActivity to be pushed onto an independent task stack, enabling normal navigation back to the mini program page. **Impact:** 1. AdMob does not add FLAG_ACTIVITY_NEW_TASK when displaying rewarded video ads, so those ads are unaffected. 2. Banner ad target pages will have a separate task stack, which may affect the final return logic if users switch tasks through the recent tasks list.	Removable, but removing it will prevent normal navigation back to the mini program page from the banner ad target page.
android:excludeFromRecents="true"	**Purpose**:	Removable, but removing it may result in a black screen task

	Since AdActivity may only serve as a transitional page, it could display a black screen task in the recent tasks list when pushed onto a new task stack. This attribute prevents the display of that black screen task. **Impact:** If AdActivity is the root page of the task stack, it will not be shown in the recent tasks list, which may lead to activity leaks in some scenarios.	being displayed in the recent tasks list.

Preconfigure Offline Mini Programs

Last updated：2025-06-06 18:27:19

Offline mini programs are embedded within your superapp. You need to download the mini program package from the console and import it into the superapp project, bundling it with the app. Users can open and run these mini programs without downloading them from the backend, even without an internet connection.

Steps

1. Download the required mini program from the console.

2. Copy the downloaded mini program package to a custom assets directory. Follow strict naming conventions and do not change it.

> Naming conventions: `{miniAppId}_{miniAppVersion}.apkg`

3. Specify the assets directory containing the offline mini programs in the SDK initialization configuration.

Notes

Offline mini programs must follow the standard binding and publishing process. Only published miniprograms can be downloaded as offline packages;
Offline mini programs adhere to version management logic, including new releases, version rollbacks, and deprecations. If the online version differs from the preloaded version, the client will fetch the online version;
If a mini program is deprecated, the corresponding offline mini program in the app will also become unusable.

Android SDK Description

Last updated：2025-06-06 18:27:19

Why use multi-process for Android SDK?

The Android runtime SDK uses a multi-process mechanism, where the mini-program and the superapp run in separate processes. The processes do not interfere with each other and transfer data through cross-process communication. This approach offers several benefits:
Does not occupy the memory of the superapp The system allocates separate memory for the mini program process, preventing memory overflow issues in the superapp.
Ensure safe and stable operation of the superapp. Any exceptions in the mini program process do not affect the superapp, ensuring continuous operation even if the mini program crashes.
Data decoupling Processes are naturally isolated in memory, preventing direct data access between processes and avoiding data coupling issues.

How is multi-process implemented?

Android does not provide APIs for dynamic process creation. Instead, processes are created by binding them to Activities using the android:process attribute in AndroidManifest.xml. When an Activity is first launched, the system creates a new process and assigns the Activity to it.
To enhance user experience, different mini programs are displayed in separate positions in the recent tasks list. This is achieved by setting the android:taskAffinity attribute in AndroidManifest.xml, placing Activities of different mini-program processes in separate task stacks.

How to manage mini program processes?

The SDK can create up to 5 mini program processes simultaneously. If the number of mini program processes is below this limit, each new mini program will start in a new process. Once the limit is reached, any additional mini programs will reuse existing processes.
When opening the same mini program, the SDK first checks if it is already running. If it is, the corresponding process is reactivated.
The main process handles the creation, reuse, and activation of mini program processes, managing all these operations centrally.

How does mini program processes communicate with the main process?

Bidirectional communication between mini program processes and the superapp process is achieved using AIDL (Android Interface Definition Language). Both the mini program and main processes bind to the same remote service. Through the Binder mechanism, they obtain a proxy object for the service, which allows them to call AIDL interfaces for inter-process communication.

How to ensure process and thread security

Currently, all resources that need to be written during the mini program's runtime are stored in the application's sandbox directory specific to the mini program. This setup avoids writing to globally shared resources, thus eliminating inter-process competition for the same system resources. For potential process security issues, we use process-safe methods for reading and writing data during development, such as file locks for file access and ContentProvider instead of SharedPreferences for data sharing.

There are no direct thread security issues between processes. Thread security concerns arise only when different mini program processes concurrently access the main process's memory via inter-process communication or when there is concurrent read/write access within a process. The SDK addresses these thread security issues using locks, synchronized methods, and Java concurrency utilities.

Android FAQs

Last updated：2025-06-06 18:27:19

Issue: Mini program fails to start

There may be several reasons for this:
Incorrect configuration file path: The configAssetName should specify the complete path of the file in the assets directory. If the configuration file is in a subdirectory, you need to include the directory path, for example: server/tcmpp-android-configurations.json.
Modification of configuration file: You cannot modify the contents of the mini program configuration file; otherwise, the mini program will not function properly.
Mismatch in packageName: The **packageName** in the configuration file must match the superapp's **applicationId** . If they do not match, the superapp will fail to run, as the **packageName** is verified during initialization. You can disable package name verification with the following setting:

```
MiniInitConfig config = builder
         .verifyPkg(false) // Ignore the package name verification
         .build();
```

Initialization annotation: Ensure that the initialization annotation @ProxyService has generated the class ExtProxyServiceScope .

How does the SDK ensure privacy compliance?

The mini program SDK initializes when the developer calls methods from the TmfMiniSDK class. Therefore, you should call these methods only after the user has given consent for privacy compliance.

How to troubleshoot errors in custom mini program APIs?

Check if XxxJsPluginScope has been generated in the compilation path, as shown below:

Ensure that the event names defined on the client side match the method names called in the mini program.**Note that the event names for custom mini program APIs are case-sensitive.**.

Mini program domain and privacy API verification logic

While using the mini program, the legality of the API request domain name will be verified, and authorization for privacy APIs will be checked if set in the management backend. However, verification is skipped in the following scenario:
The mini program is running in an unreleased version and the mini program debugging is enabled. See Mini program debugging.

Support for modular projects

When developers use the annotations @JsPlugin or @ProxyService in multiple modules within a modular project, the following error may occur during the Make Project process:

To support multi-module projects, follow these configurations:
1. In the build.gradle of each module that uses the @JsPlugin or @ProxyService annotation, add the following code:

```
android {
    defaultConfig {
        javaCompileOptions {
            annotationProcessorOptions {
                // Configure module name: Define a unique name following Android cl
                arguments = [tcmppModuleName: "Demo"]
            }
        }
    }
}
```

2. Register the module in the initialization code:

```
@ProxyService(proxy = MiniConfigProxy.class)
public class MiniConfigProxyImpl extends MiniConfigProxy {
    @Override
    public MiniInitConfig buildConfig() {
        MiniInitConfig.Builder builder = new MiniInitConfig.Builder();

        // Register all defined modules; the registerModule parameter value should
        return builder
```

How to view SDK log output?

1. Developers can filter SDK logs in the Android Studio using the keyword TMF_MINI.

2. View mini program JS error logs.

Method 1: Filter JS logs using the keyword MINI_JS_LOG.

Method 2: Debug mini program JS errors in Chrome.

How to enable mini program debugging mode?

To facilitate debugging and log viewing for the mini program, the client needs to enable the mini program debug entry. See Custom Mini Program SDK APIs - Customizing capsule button feature

```
/**
 * Return the buttons in the "More" panel. Each button must be assigned an ID, and
 * Note: This method is called in the mini program process.
 * @param builder
 * @return
 */
public ArrayList<MoreItem> getMoreItems(MoreItemList.Builder builder){
  builder.addDebug("Debug", icon) // Set the Debug button on the control panel
  return builder.build();
}
```

Once set up, a Debug button will appear in the mini program control panel. Tapping it will enable debug mode, and you'll need to restart the mini program to activate it.

Does the mini program SDK automatically clear mini program package cache?

No, developers need to manually call the mini program SDK's deletion methods to clear the mini programs package cache (Deleting mini programs). If not, the superapp's cache will increase as the number of mini program used increases.

Why can't I use the mini program normally after enabling the R8 full mode?

R8 full mode obfuscation uses stricter rules, which can cause some annotations in older versions of the mini-program SDK to be lost, potentially preventing the mini-program from starting correctly. Starting from Android Gradle Plugin version 8.0, R8 full mode is enforced during compilation. Please upgrade the mini program SDK mini_core to version 2.0.5 or later to be compatible with the R8 full mode.

How to resolve the WebView multi-process data directory conflicts?

Issue description:

WebView does not support using the same data directory in multiple processes. If not managed properly, this can lead to the following error:

The mini program operates on a multi-process architecture, and the SDK has been adapted to address this issue. However, if customers integrate third-party libraries within the mini program process, some of these libraries may not have adapted to the WebView data directory, which can prevent the mini program from starting properly.

Solution:

1. It is recommended to avoid integrating third-party libraries in the mini program process if it is not required. The mini program process can be identified as follows:

```
public class TCMPPDemoApplication extends Application
    @Override
    public void onCreate() {
        super.onCreate();
        if (TmfMiniSDK.isMiniProcess(this)) {
            // Skip the third-party framework initialization in the TCSAS mini prog
        }
    }
}
```

2. If you must integrate third-party libraries that use WebView, modify the WebView data directory as early as possible to avoid issues. Here's how:

```
public class TCMPPDemoApplication extends Application {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        sApp = this;
        // com.tencent.tmfmini.sdk.launcher.DFMWebViewCompat;
        DFMWebViewCompat.setDataDirectorySuffix(this);
    }
}
```

Some devices display a brief black screen during cold start of mini programs

Some devices may experience a brief black screen when cold starting the mini program due to performance issues or lengthy UI thread operations. To avoid this, you can set a preview background for the proxy Activity:

1. Extend the MiniApp class to create a custom theme that modifies the default preview background to replace the black screen:

```
 <style name="MyMiniApp" parent="MiniApp">
      <!--Enable window preview using windowDisablePreview-->
      <item name="android:windowDisablePreview">false</item>
      <!--Set the preview background using windowBackground-->
      <item name="android:windowBackground">#FF0000</item>
  </style>
```

2. Update AndroidManifest.xml: Change the theme of MiniActivity to your custom theme.

```
<activity
    tools:replace="android:theme"
    android:name="com.tencent.tmfmini.sdk.ui.MiniActivity1"
    android:theme="@style/MyMiniApp" />

<activity
    tools:replace="android:theme"
    android:name="com.tencent.tmfmini.sdk.ui.MiniActivity2"
    android:theme="@style/MyMiniApp"/>

<activity
    tools:replace="android:theme"
    android:name="com.tencent.tmfmini.sdk.ui.MiniActivity3"
    android:theme="@style/MyMiniApp" />

<activity
    tools:replace="android:theme"
    android:name="com.tencent.tmfmini.sdk.ui.MiniActivity4"
    android:theme="@style/MyMiniApp"/>

<activity
    tools:replace="android:theme"
    android:name="com.tencent.tmfmini.sdk.ui.MiniActivity5"
    android:theme="@style/MyMiniApp" />

<activity
    tools:replace="android:theme"
    android:name="com.tencent.tmfmini.sdk.ui.MiniActivity6"
    android:theme="@style/MyMiniApp" />
```

Android Error Codes

Last updated：2025-02-24 19:34:38

Error code	Description
0	Successful
-11001	Shark network error
-11002	Server returned an error code
-11003	Server response is empty
-11004	Server returned an invalid mini program update type
-11005	Server returned data parsing exception
-11006	The mini program does not exist or has been removed
-11007	Monthly active user limit reached
-11008	Resource limit reached
-11011	Single API request frequency too high
-11012	Total request frequency too high
-12001	The Shark instance is null
-12002	Previewing the mini program requires login
-12003	Data parsing exception
-12005	Missing mini program information
-12006	QR code error
-12007	Invalid TMF mini program QR code
-12008	The mini program appId is empty
-12009	The businessId is empty
-12010	Exception occurred while starting the mini program
-12011	JSON parsing error

-12012	Mini program download failed
-12013	Mini program parsing failed
-12100	MBEngine not integrated, game not supported
-12101	Mini game disabled by server
-12103	Mini game base library load failed
-12104	The game.js or the specified entry file does not exist

Android Privacy Compliance

Last updated：2025-03-20 18:08:15

Information about the system permissions involved in the mini program SDK:

SDK name	Required system permissions	Permission usage description
mini_core (including gateway)	android.permission.INTERNET android.permission.READ_EXTERNAL_STORAGE android.permission.WRITE_EXTERNAL_STORAGE android.permission.ACCESS_WIFI_STATE android.permission.ACCESS_NETWORK_STATE android.permission.WAKE_LOCK android.permission.READ_MEDIA_AUDIO android.permission.CHANGE_NETWORK_STATE	Access the internet for mini program container and backend interactions. Access local storage for data persistence.
mini_extra_qrc ode	android.permission.CAMERA android.permission.WRITE_EXTERNAL_STORAGE android.permission.FLASHLIGHT	Access the camera for QR code scanning. Access local images for QR code recognition.
mini_extra_v8	None	-
mini_extra_dynamic_x5	None	-
mini_extra_static_x5	None	-
mini_extra_static_x5_new	None	-
mini_extra_public_x5	None	-
mini_extra_map	None	-
mini_extra_google_map	android.permission.ACCESS_FINE_LOCATION	Access precise location for location services.
mini_extra_huawei_map	android.permission.INTERNET android.permission.ACCESS_NETWORK_STATE android.permission.CHANGE_WIFI_STATE android.permission.ACCESS_COARSE_LOCATION android.permission.ACCESS_FINE_LOCATION	Required for Huawei map services integration.
mini_extra_trtc_live	None	-

mini_extra_nfc	android.permission.NFC android.permission.NFC_TRANSACTION_EVENT	Required for using the device's NFC capabilities.
mini_extra_lbs	android.permission.ACCESS_FINE_LOCATION	Access device location for location-related APIs.
mini_extra_bluetooth	android.permission.BLUETOOTH_ADMIN android.permission.BLUETOOTH_SCAN android.permission.BLUETOOTH_ADVERTISE android.permission.BLUETOOTH_CONNECT	Required for Bluetooth-related API capabilities.
mini_extra_contact	android.permission.WRITE_CONTACTS android.permission.READ_CONTACTS	Read and write contacts for mini program communication-related APIs.
mini_extra_soter	android.permission.USE_FINGERPRINT android.permission.USE_BIOMETRIC	Biometric permissions for mini program biometric-related APIs.
mini_extra_clipboard	None	-
mini_extra_calendar	android.permission.READ_CALENDAR android.permission.WRITE_CALENDAR	Calendar access for mini program schedule management APIs.
mini_extra_lamemp3	None	-
mini_extra_doc	None	-
mini_extra_wifi	android.permission.ACCESS_FINE_LOCATION android.permission.ACCESS_WIFI_STATE android.permission.CHANGE_WIFI_STATE android.permission.ACCESS_NETWORK_STATE android.permission.CHANGE_NETWORK_STATE	Network state and modification permissions for mini program Wi-Fi management APIs.
mini_extra_network	android.permission.INTERNET android.permission.ACCESS_NETWORK_STATE	Network and network state permissions for mini program network-related APIs.
mini_extra_media_support	None	-
tbscore	android.permission.INTERNET	-

dynamicx5	android.permission.INTERNET	-
com.tencent.tbs:tbssdk	Provided by third-parties, unknown.	-
tbs-doc-support	Provided by third-parties, unknown.	-

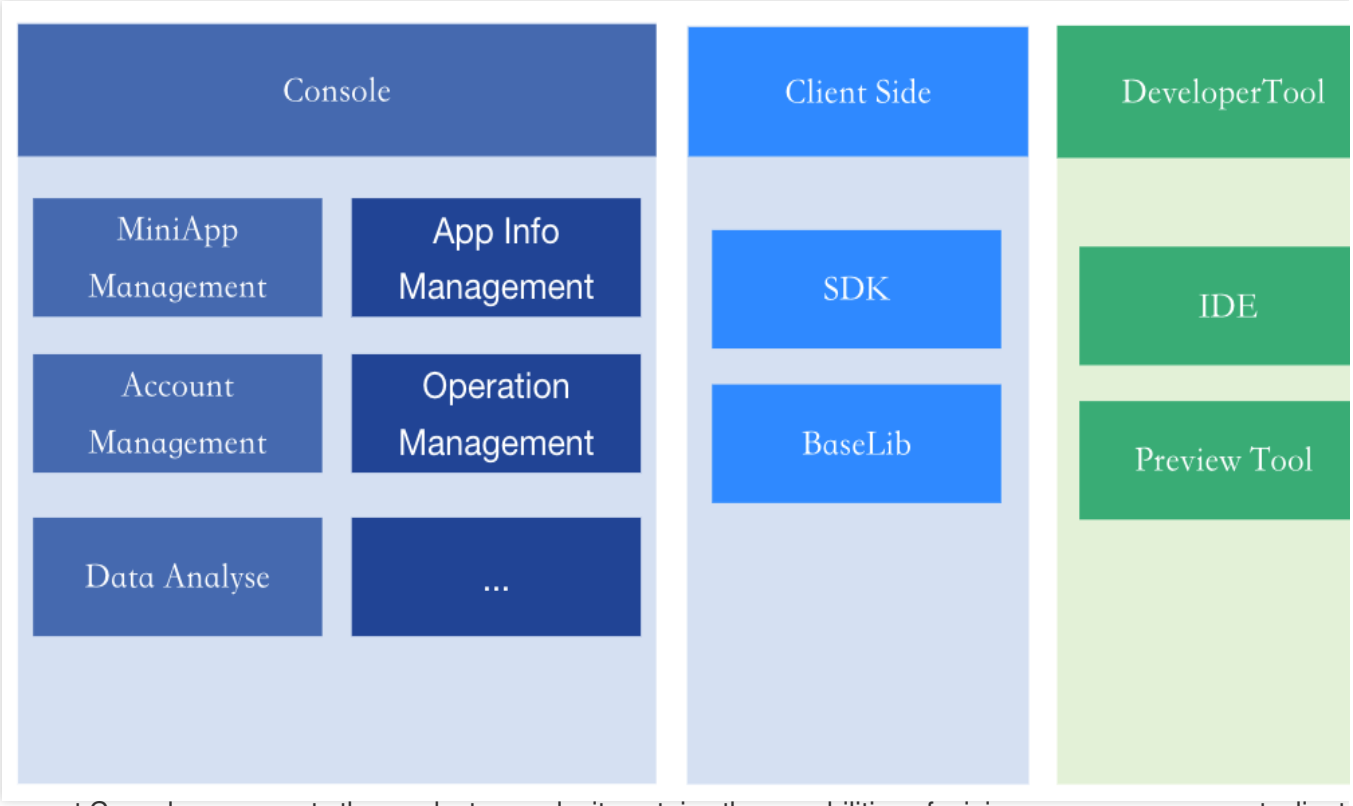
For more information, see Privacy Compliance.

iOS
iOS SDK Description
SDK Introduction

Last updated：2025-02-10 16:29:08

Product Introduction

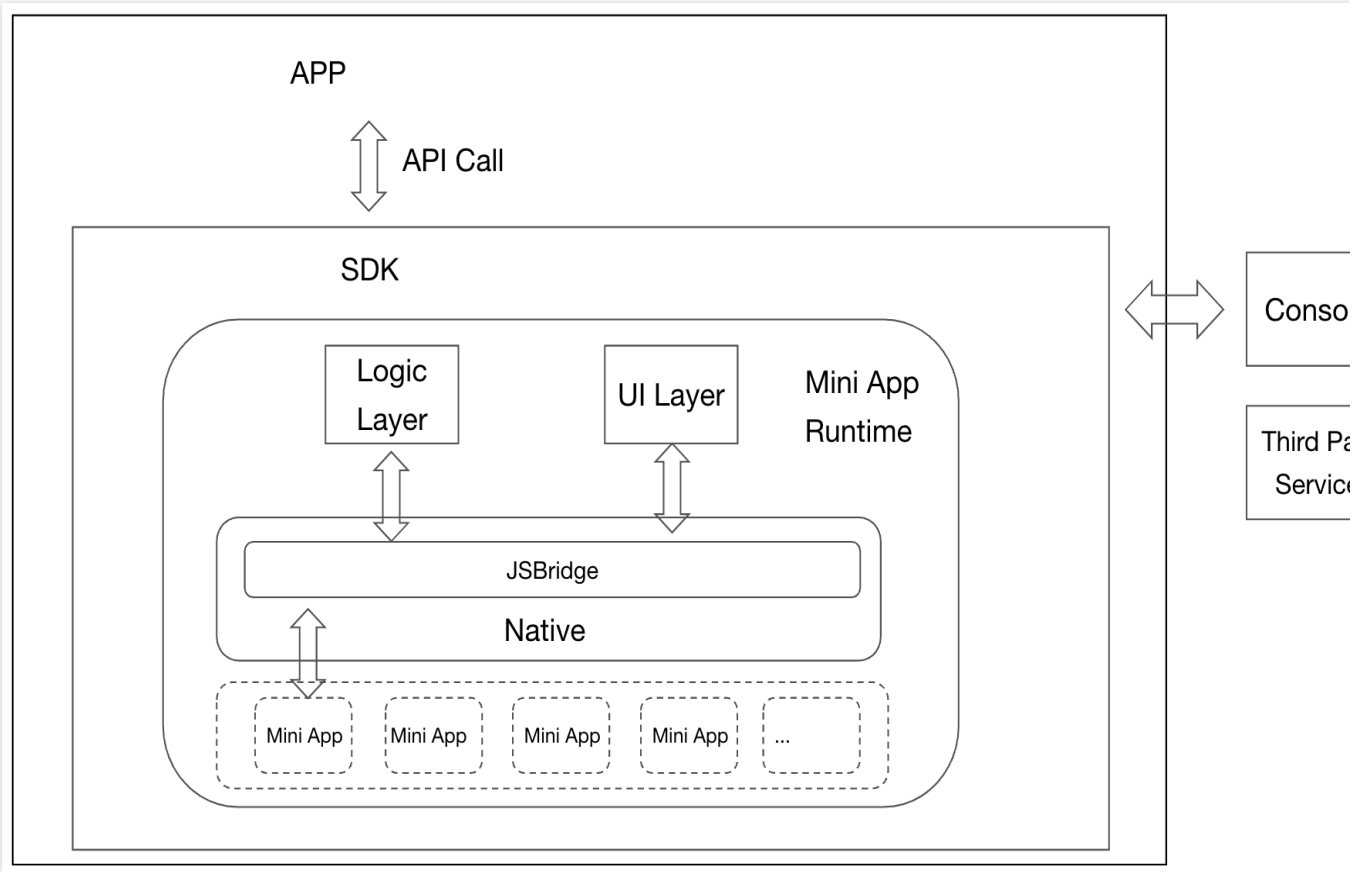
The product consists of three parts: Management Console, Client SDK and mini program development tool (IDE).



Management Console: represents the product console; it contains the capabilities of mini program management, client application management, mini program operation and maintenance management, and so on.
Client: represents the mobile client application that integrates mini program SDK; mini program SDK provides the runtime environment of mini program for the client application.
Developer tools: mini program developers use mini program IDE for mini program development, debugging and version submission; mini program developers use mini program preview assistant to preview and verify the mini program in the mobile client.

Mini Program SDK Working Principle

The mini program SDK provides the runtime environment for the mini program to the client application; the mini program SDK communicates with the product backend to get the information of the mini program, and loads and runs the mini program in the runtime environment provided by the mini program SDK.



Mini program SDK is divided into core library (i.e. TCMPPSDK) and extension library (i.e. TCMPPExtXXX) for the consideration of reducing package size and system privilege and function control, most of the functions can be realised by integrating only the core library, and there is no need to apply for extra system privilege. The impact of integrating only the core library on the final app installation package is around 4M.
Please refer to iOS Extended component for the description of the Extension Library.

SDK Quick Integration

Last updated：2025-06-12 15:54:52

Example for integration

You can get the demo address from GitHub.

Prerequisites

Environment requirements

iOS >= 9.0

Xcode >= 10.0

Component dependency

tars

MQQComponents

TMFShark

SSZipArchive

PromiseObjC

MJRefresh

SocketRocket

Brotli

CocoaAsyncSocket

Integration method

TCMPPSDK can be integrated in the following two ways:

CocoaPods integration of SDK

Manual integration of SDK

CocoaPods integration of SDK

1. Add the source and mini program dependency modules to the Podfile in your project:

```
# Pods repository
source 'https://e.coding.net/tcmpp-work/tcmpp/tcmpp-repo.git'

target 'YourTarget' do
```

```
    # ---- TCSAS ------------------------------------------------------------ #
    pod 'TCMPPSDK'
    pod 'TCMPPExtScanCode'
    pod 'TCMPPExtMedia'

end
```

In this case, `Replace YourTarget` with the name of the target in your project that requires the `SDK` .

2. Run cd on the Terminal to go to the Podfile directory and run pod install to install the component.

```
$ pod install
```

Note:

If you encounter the error `Couldn't determine repo type for URL: '` https://e.coding.net/tcmpp-work/tcmpp/tcmpp-repo.git `':` you need to run the following command before executing `pod install` : `pod repo add specs` https://e.coding.net/tcmpp-work/tcmpp/tcmpp-repo.git.

Manual integration of SDK

Example for integration

You can click here to download the manual integration demo.

1. **Add the SDK**

Add the directory of the SDK components to the appropriate location in your Xcode project and select the relevant target.

You can quickly add the components by dragging the directory directly from Finder into your Xcode project.

2. **Add dependent SDKs**

Add all components depended on by SDK to the project. For the list of dependent components, see Component dependency.

3. **Add dependent system libraries**

Add the system libraries that the SDK depends on to the project, open the project settings page in Xcode, select the target, and click **General** to add them under Linked Frameworks and Libraries.

4. **The system library dependencies are as follows:**

Foundation.framework

CoreTelephony.framework

CFNetwork.framework
Security.framework
SystemConfiguration.framework
CoreService.framework
CoreFoundation.framework
libz.tdb
libc++.tbd
libc.tbd
libbz2.tbd
libsqlite3.0.tbd

5. **Project settings**

After adding the SDK, you need to make the following project settings in Xcode:

Select **Build Settings** > **Linking** > **Other Linker Flags**, and add `-ObjC` .

Select **Build Settings > Apple Clang - Custom Compiler Flags > Other C Flags**, and add

```
-fshort-wchar
-D__FIXWCHART__
```

Select **Build Phases > Copy Bundle Resources**, and add the SDK resource file:

Integrate SDK into the Swift project

> **Example for integration**
>
> You can click here to download the Swift project integration demo.

```
import TCMPPSDK
```

Obtain the configuration file

The initialization of the mini program SDK relies on the configuration file obtained from the mini program console. Before integrating the mini program SDK, you need to get this configuration file.

Log in to the console, and click Create superapp.

Fill in the superapp information.

Download the configuration file.

Note:

The default name of the downloaded configuration file is tcsas-ios-configurations.json.

Add the configuration file to the project

After obtaining the configuration file, you need to add it to your superapp's source code.

Note:

The bundleId of the iOS project must match the bundle ID in the configuration file. If they do not match, the mini program SDK will fail to validate the package name at runtime, leading to initialization errors.

If they do not match, you cannot directly modify the bundleId field in the configuration file. Instead, use one of the following methods to correct it:

Change the bundleId in the project to match the one in the configuration file.

Modify the bundleId in the console to match the one in your project and re-download the configuration file.

The packageName field in the configuration file must be consistent with the package name of the current project.

Add permission settings

If you have only integrated the core library of the SDK, namely TCMPPSDK, you need to add permission request information in the info.plist file.

The permissions involved with the core SDK (TCMPPSDK) are as follows:

Permission	Corresponding KEY	Involved API
Photo library write	NSPhotoLibraryAddUsageDescription	saveImageToPhotosAlbum、saveVideoToPhotosAlbum
Camera	NSCameraUsageDescription	CameraContext (Camera component)
Microphone	NSMicrophoneUsageDescription	CameraContext (Camera component)

If you need to use the extended SDK, you must also add the corresponding permission request information in the project's info.plist file.

For details, refer to Extended Component SDK .

Configure to support landscape mode

The loading page in the SDK, video components, etc., support landscape mode, but the superapp project must also support it.
Please check the following option in Xcode settings:

Note:
If landscape mode is not enabled, the landscape features in the SDK will not work.

Import the header file

Import the header file into AppDelegate.

```
//TCSAS
#import <TCMPPSDK/TCMPPSDK.h>
```

Set the configuration information

Initialize the TMAServerConfig object based on the configuration file, and use it to initialize the mini program engine. The SDK supports direct engine initialization, preparing network connections in advance, and updating base library and configuration information to speed up mini program loading. It can also support initialization when needed.
Example:

```
 // Configure the environment
 NSString *filePath = [[NSBundle mainBundle] pathForResource:@"tcsas-ios-configurat
 if(filePath) {
   TMAServerConfig *config  = [[TMAServerConfig alloc] initWithFile:filePath];
   // Direct initialization
   [[TMFMiniAppSDKManager sharedInstance] setConfiguration:config];
}
```

Other initialization operations

You can set up the implementation instance of the open API as needed. If integrating extended modules, prepare the initialization of the extended API.

```
// Set the mini program engine delegate
[TMFMiniAppSDKManager sharedInstance].miniAppSdkDelegate = [MIniAppDemoSDKDelegateI
```

MIniAppDemoSDKDelegateImpl must implement the **TMFMiniAppSDKDelegate** protocol. You can refer to Introduction and the MIniAppDemoSDKDelegateImpl file in the Demo project.

Open a mini program

You can directly call the API to open the mini program using the appid:

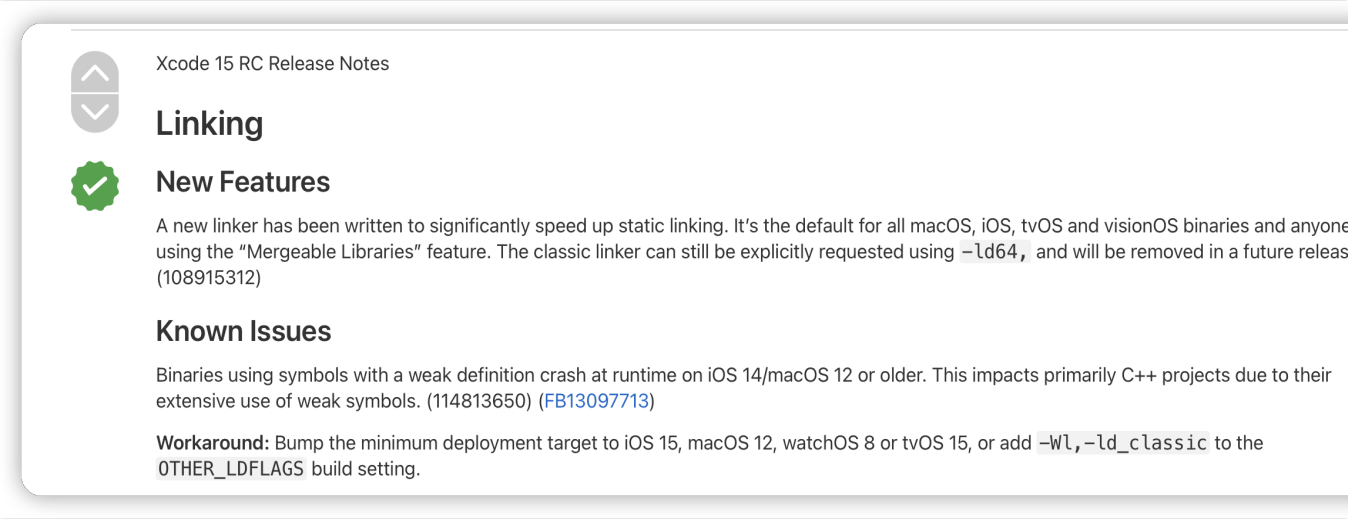
```
[[TMFMiniAppSDKManager sharedInstance] startUpMiniAppWithAppID:appId parentVC:self
   NSLog(@"open applet error:%@",error);
}];
```

SDK Integration FAQs

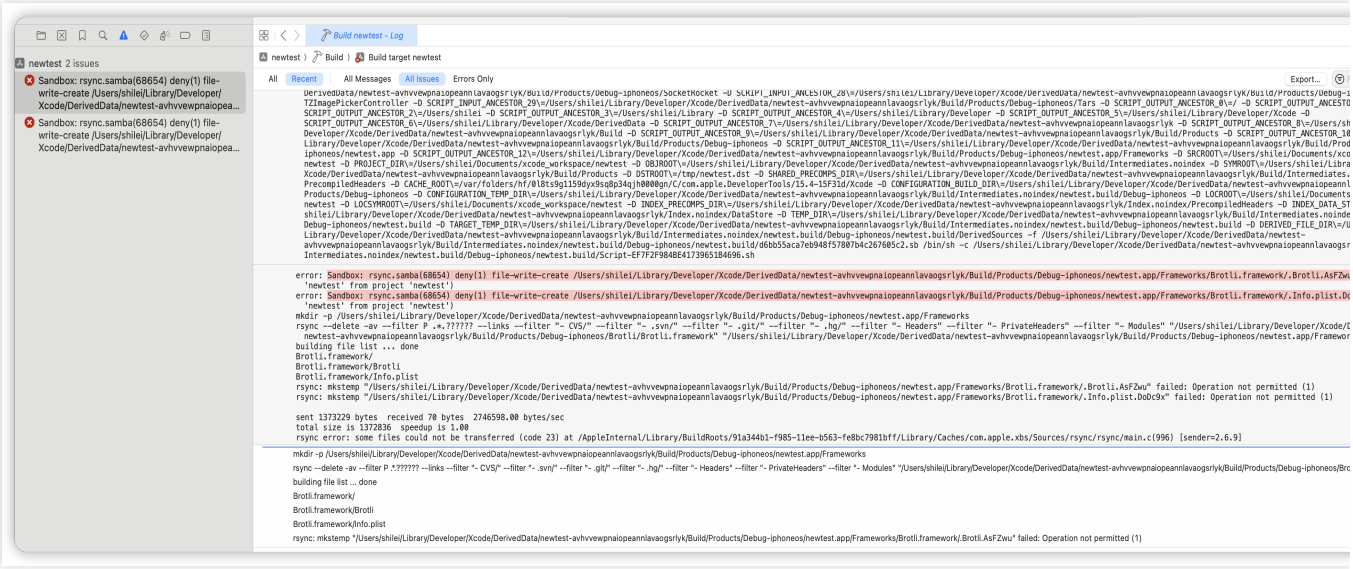
Last updated：2025-03-17 17:50:09

For XCode 15 and earlier versions, the mini program crashes on iOS 12 and later versions

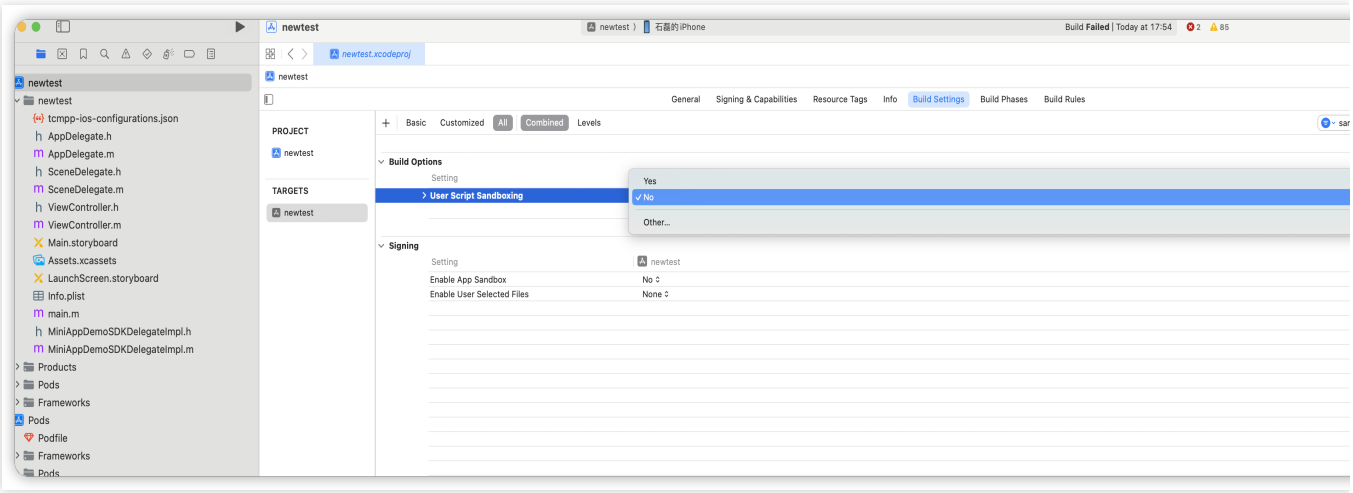
Add `-Wl,-ld_classic` to the project's compilation options.



Xcode15 compiles a new project and introduces the pod library with the error Sandbox: rsync.sanba deny(1) file-write-create xxx



Solution: Build Settings search for sandbox, change User Script Sandboxing in Build Options to NO.



Apple M-Series Chip PC Emulator runs abnormally

The current version of the SDK needs to be run through Rosetta for running on M-Series chip PC emulators.

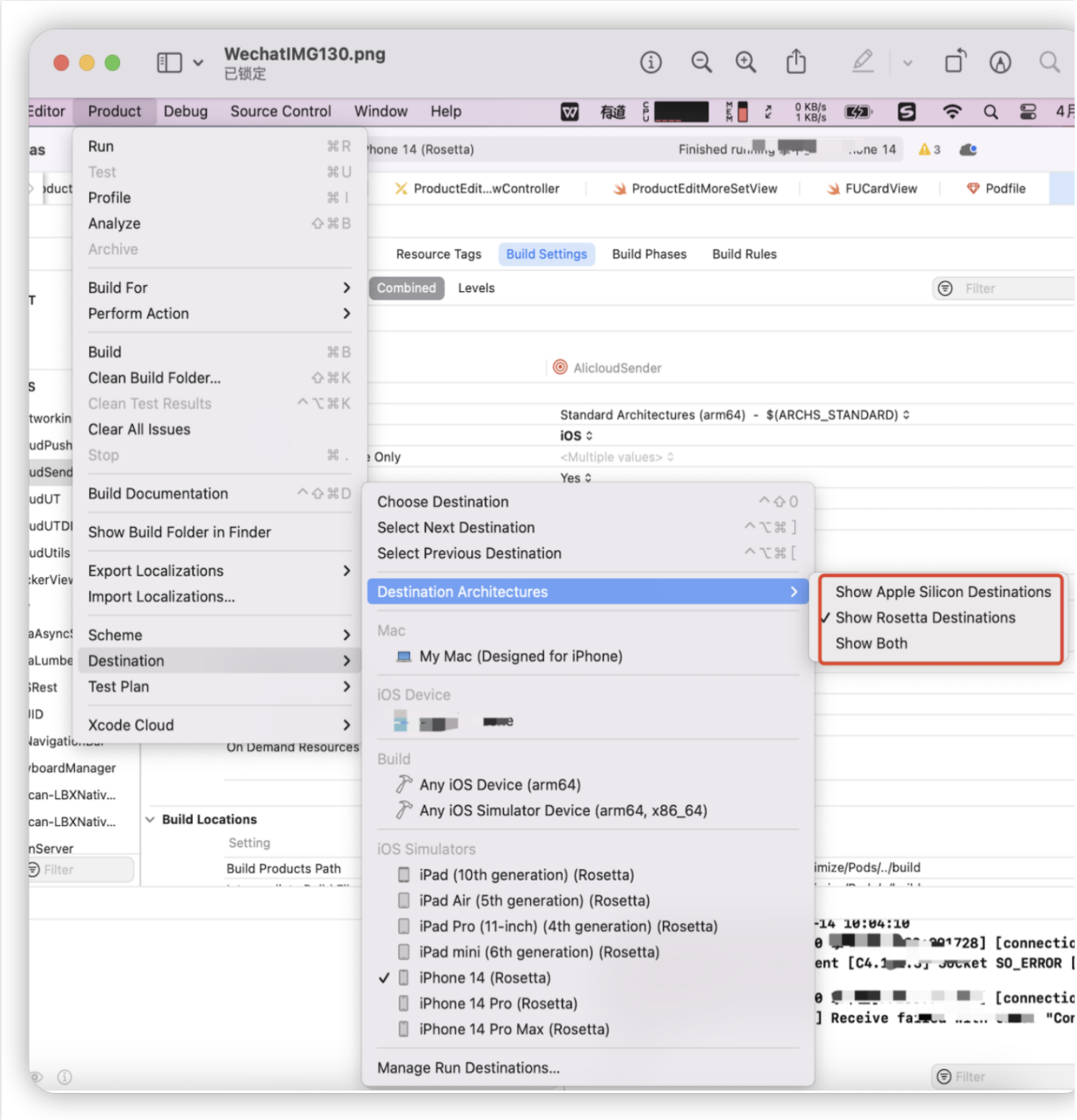
Before xcode14:

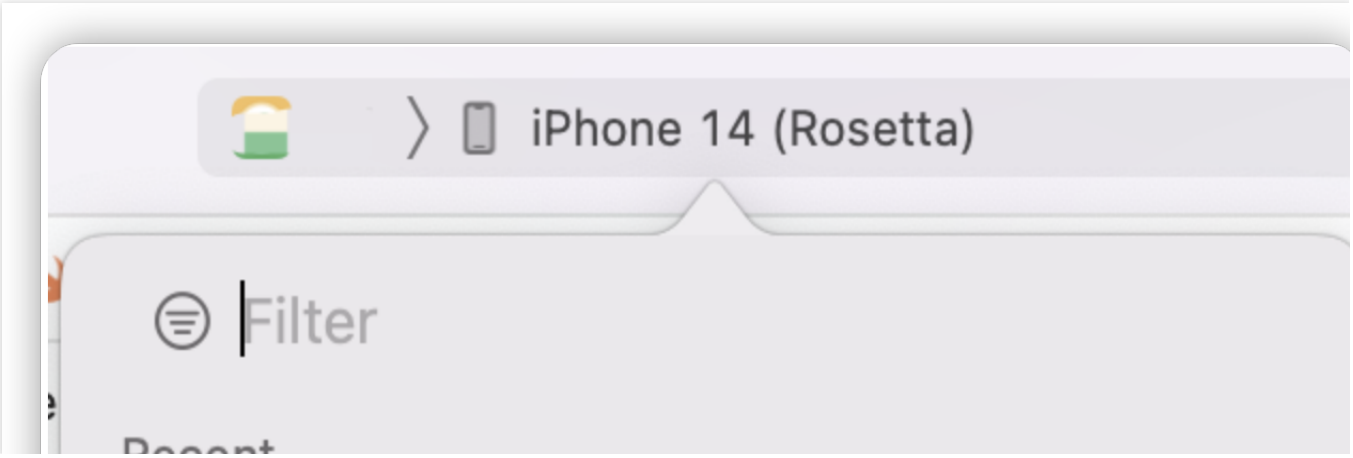
We can right click xcode->show profile->check open with Rosetta, so it is running on the emulator.

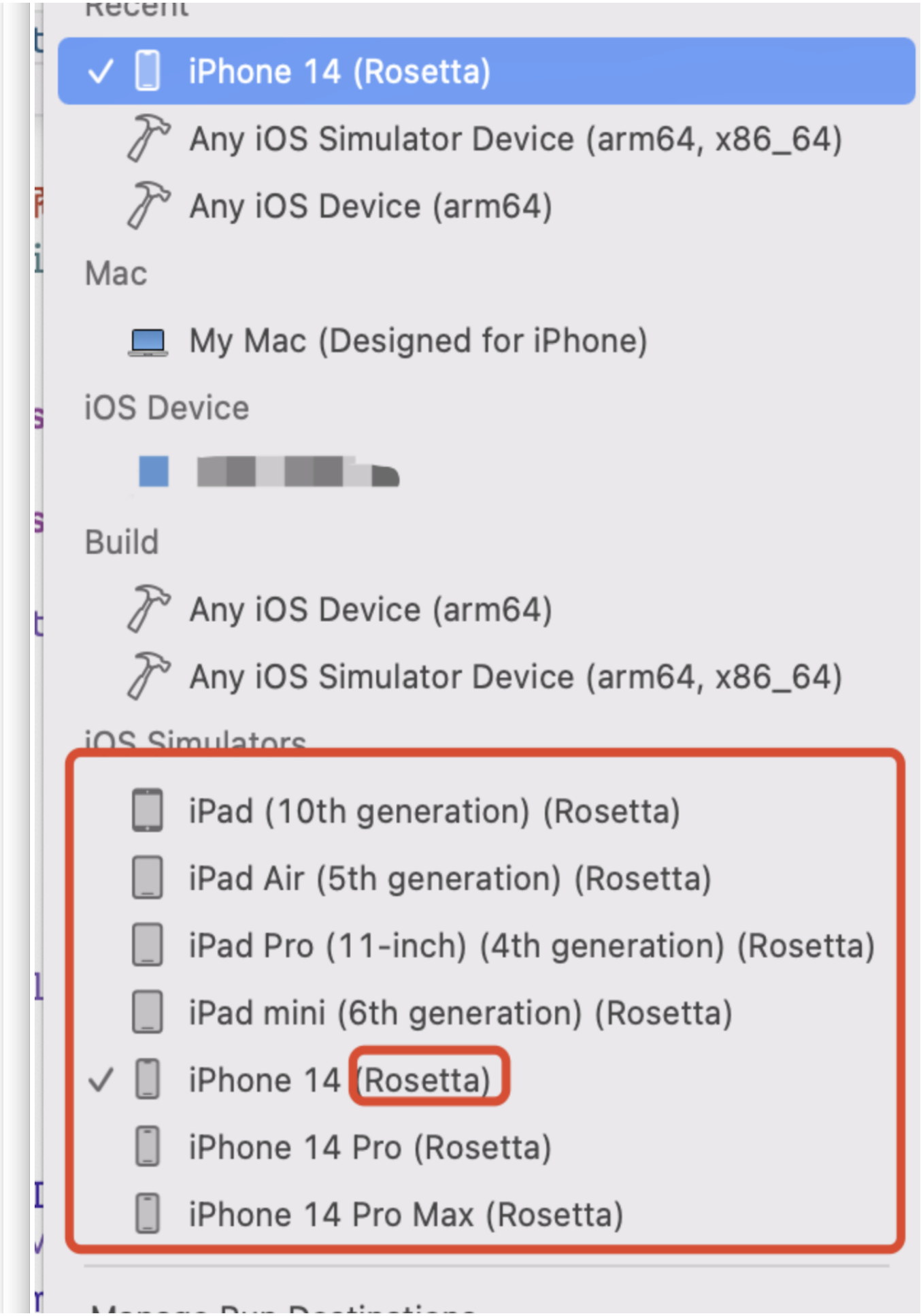
After xcode14:

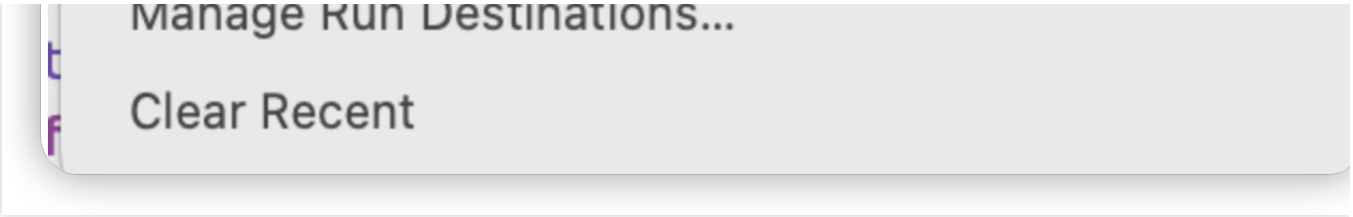
xcode Open Project > Product > Destination > Destination Architectures and you can choose which mode of simulator to open it with.

We will choose the emulator ending with (Rosetta).


WechatIMG130.png
已锁定
Editor
Product
Debug
Source Control
Window
Help
Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For
Perform Action
Build ⌘B
Clean Build Folder... ⇧⌘K
Clean Test Results ^⌥⌘K
Clear All Issues
Stop ⌘.
Build Documentation ^⇧⌘D
Show Build Folder in Finder
Export Localizations
Import Localizations...
Scheme
Destination
Test Plan
Xcode Cloud
Choose Destination ^⇧0
Select Next Destination ^⌥⌘]
Select Previous Destination ^⌥⌘[
Destination Architectures
Show Apple Silicon Destinations
Show Rosetta Destinations
Show Both
Mac
My Mac (Designed for iPhone)
iOS Device
Build
Any iOS Device (arm64)
Any iOS Simulator Device (arm64, x86_64)
iOS Simulators
iPad (10th generation) (Rosetta)
iPad Air (5th generation) (Rosetta)
iPad Pro (11-inch) (4th generation) (Rosetta)
iPad mini (6th generation) (Rosetta)
iPhone 14 (Rosetta)
iPhone 14 Pro (Rosetta)
iPhone 14 Pro Max (Rosetta)
Manage Run Destinations...
Resource Tags
Build Settings
Build Phases
Build Rules
Combined
Levels
Build Locations
Setting
Build Products Path


iPhone 14 (Rosetta)
Filter


iPhone 14 (Rosetta)
Any iOS Simulator Device (arm64, x86_64)
Any iOS Device (arm64)
Mac
My Mac (Designed for iPhone)
iOS Device
Build
Any iOS Device (arm64)
Any iOS Simulator Device (arm64, x86_64)
iPad (10th generation) (Rosetta)
iPad Air (5th generation) (Rosetta)
iPad Pro (11-inch) (4th generation) (Rosetta)
iPad mini (6th generation) (Rosetta)
iPhone 14 (Rosetta)
iPhone 14 Pro (Rosetta)
iPhone 14 Pro Max (Rosetta)



When issues occur during download, opening, or searching for mini programs, the SDK will throw error codes. These codes help identify whether the issue is with the gateway, backend, or SDK.

Gateway errors

If the logs contain TMFSharkXXX related errors, the issue is with the gateway. Common errors include:
TMFSharkTaskErrorDomain
TMFSharkDataAccessLayerErrorDomain
TMFSharkRequestErrorDomain
For details about the error codes, see iOS Error Codes.

Backend service errors

If the logs contain TMAMiniAppErrorDomain related errors, the issue is likely with the mini program backend. Common errors include:
TMAMiniAppErrorFileMd5VerifyFail - The MD5 verification of the mini program package failed.
TMAMiniAppErrorConfigInitError - Incorrect initialization information or uninitialized. Check the configuration file.
TMAMiniAppErrorJSSDKDownloadFail - Mini program base library download failed.
TMAMiniAppErrorErrorLink - Incorrect mini program link.
For details about the error codes, see iOS Error Codes.

SDK errors

If the logs contain MFMiniApp Error but do not include TMFSharkXXX or TMAMiniAppErrorDomain errors, the issue may be related to console configurations. Common issues include:
Domain name blocklist: appID(appName)skipDomainCheck,but url:(domainUrl)is in black list. Search "skipDomainCheck,but url:"in the log to confirm the issue.
Domain name allowlist: appID(appName) check url in domainList, result is 0. Search "in domainList, result is " to confirm the issue.
Privacy API: If calling sensitive APIs (e.g., chooseLocation, getLocation, choosePoi) and the console has related permissions configured, you might see errors like PluginEngine |eventName(chooseLocation) fail. no permission.

Ensure the mini program has the necessary permissions enabled.

iOS API
Mini Program Management APIs
Opening Mini Programs

Last updated：2025-05-26 15:24:12

Opening mini programs

When opening a mini program, the system will check whether there is a locally cached one. If not, it will automatically download the mini program from the remote server and then open it. If a cached version is available, the local mini program will be opened first while the system checks in the background for any new versions on the server.

Notes:

If a new version is available, it will be downloaded, and the next time you open the mini program, the updated version will be used.

```
// Open a mini app through the mini program ID

// @param appID - Mini program ID
// @param verType - The version type of the mini program to open
// @param scene - Scene value
// @param firstPage - The first page of the mini program to open
// @param paramsStr - The parameter used to open the mini program
// @param parentVC - The first view controller to call
// @param completion - Error callback
- (void)startUpMiniAppWithAppID:(NSString *)appID
                        verType:(TMAVersionType)verType
                          scene:(TMAEntryScene)scene
                      firstPage:(NSString * _Nullable)firstPage
                      paramsStr:(NSString * _Nullable)paramsStr
                       parentVC:(UIViewController *)parentVC
                     completion:(void (^)(NSError * _Nullable))completion;
```

Parameters supported by options:

Name	Required	Type	Description
appID	YES	NSString	ID of the mini program to open
verType	YES	TMAVersionType	Type of the mini program to open
scene	YES	TMAEntryScene	The scene value for opening the mini program

firstPage	NO	NSString	The first page of the mini program to open
paramsStr	NO	NSString	The parameter used to open the mini program
parentVC	YES	UIViewController	The first view controller to call
completion	YES	block	Error callback

Opening a mini program by mini program ID (appId)

To open the released version of a mini program :

```
[[TMFMiniAppSDKManager sharedInstance] startUpMiniAppWithAppID:appId parentVC:self
   NSLog(@"open applet error:%@",error);
}];
```

Notes:

The appId field is the ID of the mini program, which can be obtained from the mini program developer or via the mini program search API.

To open the Preview or development version of a mini program:

```
[[TMFMiniAppSDKManager sharedInstance] startUpMiniAppWithAppID:appId verType:verTyp
   NSLog(@"open applet error:%@",error);
}];
```

Notes:

The appVerType should match the version of the mini program. The value of the appVerType can be obtained from the TMFAppletSearchInfo object instance returned by the API (getRecentList).

The value of appVerType for the Preview of the mini program should be TMAVersionPreview.

The value of appVerType for the development version of the mini program should be TMAVersionDevelop.

Calling QRCode API to open a mini program

The mini program SDK provides a capability to open mini programs by calling the QRCode API. You need to integrate with the extension library TCMPPExtScanCode before calling the API.

```
// Open the mini program through QRCode
// @param parentVC - The first view controller to call
// @param completion - Error callback
- (void)startUpMiniAppWithQRCodeWithParentVC:(UIViewController *)parentVC
                                  completion:(void (^)(NSError * _Nullable))complet
```

Opening a mini program through QR code

The mini program SDK provides an API to open mini programs based on QR code scanning in the console. Before using the scanning capability provided by the mini program SDK, you need to integrate the scanning extension capability. For details about the integration, refer to the scanning capability documentation.
After integrating the scanning capability, you can **start the QR code scanning and open the mini program as follows:**

```
/// Open a mini program via the QR code
/// @param qrData - QR code content
/// @param parentVC - The first view controller to call
/// @param completion - Error callback
- (void)startUpMiniAppWithQrData:(NSString *)qrData
                       parentVC:(UIViewController *)parentVC
                      completion:(void (^)(NSError * _Nullable error))completion;
```

Opening the released version of a mini program through QR code

Starting from version 2.0.9, the QR code for the released version of a mini program can be generated and modified in the console. Once the app scheme is configured, you can open the mini program by scanning the QR code with the system camera.
Go to **Application managemen**t - **Mini program approval** - **Submitted** in the console, click **Download QR code** in the **Operation** column, and the following pop-up appears:

The default scheme is tcmpp plus last few characters of the appid, which can be found through **Application management** - **Application list**.

For example, if the appid is app-ylk2jebx9q, then the schema is tcmppylk2jebx9q. You can click **Modify** to modify the scheme.

The scheme name is returned through the proxy API **getAppScheme** in the client. See the sample code below:

```
- (NSString *)getAppScheme{
    return @"tcmpp";
}
```

Configure the same scheme in the URL Types of the main project's info.plist file:

Implement the handleOpenUrl method in the openUrl method. See the sample code below:

```
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url options:(NSDictionary
    if ([[TMFMiniAppSDKManager sharedInstance] handleOpenUrl:url]){
        return YES;
    }
    return YES;
 }
```

Once completing the above steps, you can scan the QR code in the console with the system camera to open the released version of the mini program.

Closing Mini Programs

Last updated：2024-11-21 17:56:03

Closing mini programs

Note：

Closing a mini program means only closing its page, switching the mini program to the background. It remains in memory, allowing for a quick relaunch with minimal delay.

Note：

1. A mini program remains active in memory for 10 minutes after closing.
2. Up to 3 mini programs can remain active simultaneously.

1. Closing a specific mini program

```
/// Close a specific mini program
/// @param appID The appID of the mini program to close
- (void)closeMiniAppWithAppID:(NSString *)appID;
```

2. Closing current mini program

```
/// Close the currently running mini program
- (void)closeCurrentApplet;
```

3. Closing all running mini programs

```
/// Close all running mini programs in memory
- (void)closeAllApplications;
```

Terminating mini programs

Note：

Terminating a mini program means clearing it from memory. The next time it is opened, it will need to be reloaded into memory, which takes more time.

1. Terminating the specified mini program

```
/// Terminate the specified mini program object.
```

```
/// @param appID The appID to terminate, all versions are deleted by default.
- (void)terminateMiniAppWithAppID:(NSString *)appID;
```

2. Terminating the current mini program

```
/// Terminate the currently running mini program object.
- (void)terminateCurrentApplet.
```

3. Terminate all mini programs.

```
/// Terminate all mini programs on the stack and alive
- (void)terminateAllApplications.
```

Deleting Mini Programs

Last updated：2024-11-21 17:56:22

Note：

As the mini program runs, it caches the mini program package and mini program information locally so that it can be opened more quickly next time. If you want to delete all the information of a mini program, you can call the following API to delete a mini program or delete all mini programs.

Delete all mini program caches

```
/// Remove all caches related to mini programs including resource packs, base libra
- (void)clearMiniAppCache;
```

Delete the specified mini program cache

```
/// Delete the local cache mini program.
/// @param appID appID to be deleted, defaults to all versions.
- (void)clearCacheWithAppID:(NSString *)appID;
```

Deletes the specified version and type of mini program

```
// Delete local cache mini programs
// Delete the local cache mini program.
// @param appID appID to delete, defaults to all versions - appID to delete, defaul
// @param verType the type of local cache applet to delete - the version type of th
- (void)clearCacheWithAppID:(NSString *)appID verType:(TMAVersionType)verType;
```

Sample code:

```
[[TMFMiniAppSDKManager sharedInstance] clearCacheWithAppID:appId verType:verType];
```

Searching Mini Programs

Last updated：2025-02-24 19:34:38

The mini program SDK provides an API for searching mini programs online by keywords and categories.

Note:

The firstType and secondType parameters are used to specify the primary category and secondary category for the mini programs to be searched.

The completion parameter is used to obtain the search results of the mini programs.

```
// Search the mini program

// @param name - Keywords in the search name
// @param completion - Search results
- (void)searchAppletsWithName:(NSString *)name
                   completion:(void (^)(NSArray<TMFAppletSearchInfo *> * _Nullab

// Search the mini program

// @param name - Keywords in the search name
// @param firstType - Primary category name
// @param secondType - Secondary category name
// @param completion - Search results
- (void)searchAppletsWithName:(NSString * _Nullable )name
                    firstType:(NSString * _Nullable)firstType
                   secondType:(NSString * _Nullable)secondType
                   completion:(void (^)(NSArray<TMFAppletSearchInfo *> * _Nullable,
```

Search by keyword

Example:

```
[[TMFMiniAppSDKManager sharedInstance] searchAppletsWithName:searchString completio
    if (error) {
      // Search failed, or the list is empty
    } else {
      // Search successful, list is not empty
    }
}];
```

Search by a single category

Example:

```
[[TMFMiniAppSDKManager sharedInstance] searchAppletsWithName:nil firstType:@"firstT
```

```
    if (error) {
      // Search failed, or the list is empty
    } else {
      // Search successful, list is not empty
    }
}];
```

Search by dual categories

Note:

The result will be the intersection of the two categories.

Example:

```
[[TMFMiniAppSDKManager sharedInstance] searchAppletsWithName:nil firstType:@"firstT
    if (error) {
      // Search failed, or the list is empty
    } else {
      // Search successful, list is not empty
    }
}];
```

Specify search scope

Note:

You can specify a search for mini programs and mini games, or only mini programs or mini games.

```
/**
 Type, i.e., mini programs or mini games
- TMAAppTypeApp: Mini programs
- TMAAppTypeGame: Mini games
*/

typedef NS_ENUM(int32_t, TMASearchAppType) {
    TMASearchAppTypeAll = 0,
    TMASearchAppTypeApp = 1,
    TMASearchAppTypeGame = 2,
};

// Search the mini program

// @param name - Keywords in the search name
// @param searchType - Search scope, i.e., mini programs or mini games
// @param firstType - Primary category name
// @param secondType - Secondary category name
```

```
// @param completion - Search results
- (void)searchAppletsWithName:(NSString * _Nullable )name
                   searchType:(TMASearchAppType)searchType
                   firstType:(NSString * _Nullable)firstType
                  secondType:(NSString * _Nullable)secondType
                  completion:(void (^)(NSArray<TMFAppletSearchInfo *> * _Nullable,

// Gets the mini program category information

// @param existMnpOnly - Whether to return only mini program results
// @param searchType - Search scope, i.e., mini programs or mini games
// @param completion - The search results by category
- (void)getTypeInfoExistMnpOnly:(BOOL)existMnpOnly
                    searchType:(TMASearchAppType)searchType

WithCompletion:(void (^)(NSArray<nsdictionary><nsstring *,nsarray *> *> * _Nullable
```

Getting Recently Accessed Mini Program List and Information

Last updated：2024-11-21 17:57:24

Getting the information of all mini programs opened recently

```
// Get the information of all mini programs opened recently
///@return Mini program array<TMFMiniAppInfo>
- (NSArray *)loadAppletsFromCache;
```

Getting the information of a running mini program

```
/// Get the object of the currently running mini program
/// @return TMFAppletInfo Mini program information
- (TMFMiniAppInfo *)currentApplet;
```

Pre-Downloading Mini Programs

Last updated：2024-11-21 17:57:45

To reduce the waiting time when opening a mini program and optimize user experience, the following API is provided to pre-download mini program packages.

```
///Pre-prepare mini program information
///@param appIds  Information of mini program to prepare
///@param isDownload Whether to download directly
///@param complete Callback for batch updating mini programs
- (void)preloadMiniApps:(NSArray *)appIds isDownload:(BOOL)isDownload complete:(voi
```

Mini Program File Management

Last updated：2024-11-21 17:58:08

Converting wxfile path to absolute path

In some scenarios, you may need to convert a mini program file path to an absolute path to access the file data. For example, when using the mini program's forwarding feature, the returned image path is a mini program file path. You can convert it to an absolute path to access the image data and then initiate third-party sharing. Similarly, in custom APIs, you can pass the mini program file path as a parameter, and the host app can convert it to an absolute path to access the file data.

Sample code:

```
TMAFileManager *fm = [[TMFMiniAppSDKManager sharedInstance] getFileManagerWithAppID
NSString *filePath = [fm translateWxfilePathToAbsolutePath:wxPath];
```

Converting absolute path to wxfile path

The SDK supports converting file paths created in the mini program's cache directory to mini program file paths for internal use.

Sample code:

```
TMAFileManager *fm = [[TMFMiniAppSDKManager sharedInstance] getFileManagerWithAppID
NSString *tmpPath = [fm createMediaTmpPathWithFileName:@"a.pdf" type:TMATmpPathType

NSString *wxPath = [fm translateAnyPathToWxfilePath:tmpPath];
```

Creating files in the mini program temporary directory

The SDK supports creating files in the mini program's cache directory natively.

Sample code:

```
TMAFileManager *fm = [[TMFMiniAppSDKManager sharedInstance] getFileManagerWithAppID
NSString *tmpPath = [fm createMediaTmpPathWithFileName:@"a.pdf" type:TMATmpPathType
```

Customize Mini Program Capabilities
Custom Mini Program APIs

Last updated：2025-03-17 15:45:40

The SDK engine provides an extension mechanism that allows host apps to customise APIs for mini programs to call.

Implementation steps

1. Customise a class and import the TMAExternalJSPlugin.

```
#import <TCMPPSDK/TCMPPSDK.h>

@interface NativePluginTest : NSObject

@end
```

2. Declare TMA_REGISTER_EXTENAL_JSPLUGIN and add a custom API via TMAExternalJSAPI_IMP().

reference case

```
#import "NativePluginTest.h"
#import "TMAExternalJSPlugin.h"
#import "TMFMiniAppInfo.h" #import "TMFMiniAppInfo.h" #import "TMFMiniAppInfo.h".

Implementing NativePluginTest

TMA_register_extenal_jsplugin; //Custom sync api.

// custom sync api
TMAExternalJSAPI_IMP(testSync) {
    TMFMiniAppInfo *appInfo = context.tmfAppInfo; NSDictionary *data = context.tmfA
    NSDictionary *data = params[@"data"];

    NSLog(@"************ invokeNativePlugin testSync,appId:%@,data is %@",appInfo.a

    TMAExternalJSPluginResult *pluginResult = [TMAExternalJSPluginResult new]; [TMA
    pluginResult.result = @{}; return
    return pluginResult;
}

TMAExternalJSAPI_IMP(test) {
    TMFMiniAppInfo *appInfo = context.tmfAppInfo; TMFMiniAppInfo *appInfo = context
    NSDictionary *data = params[@"data"];
```

```
    NSLog(@"************ invokeNativePlugin test,appId:%@,data is %@",appInfo.appId

    // asynchronous processing, return the result to the mini program in an async c
    //{
    // TMAExternalJSPluginResult *pluginResult = [TMAExternalJSPluginResult new]; /
    // pluginResult.result = @{@"result": result.data}; // [context doCallback
    // [context doCallback:pluginResult].
    // }

    return nil;
}

@end
```

It can be used like this in a mini program.

```
// Asynchronous api calls
var opts = {
   api_name: 'test',
   success: function(res) {},
   failure: function(res) {},
   completion: function(res) {},
   data: { // Input
      Name : 'kka',
      age : 22
   }
 }
 wx.invokeNativePlugin(opts); // Synchronise api calls.

// Synchronise api calls
var opts = {
   api_name: 'testSync',
   sync:true
}
var rst = wx.invokeNativePlugin(opts); var rst = wx.
```

Advanced use

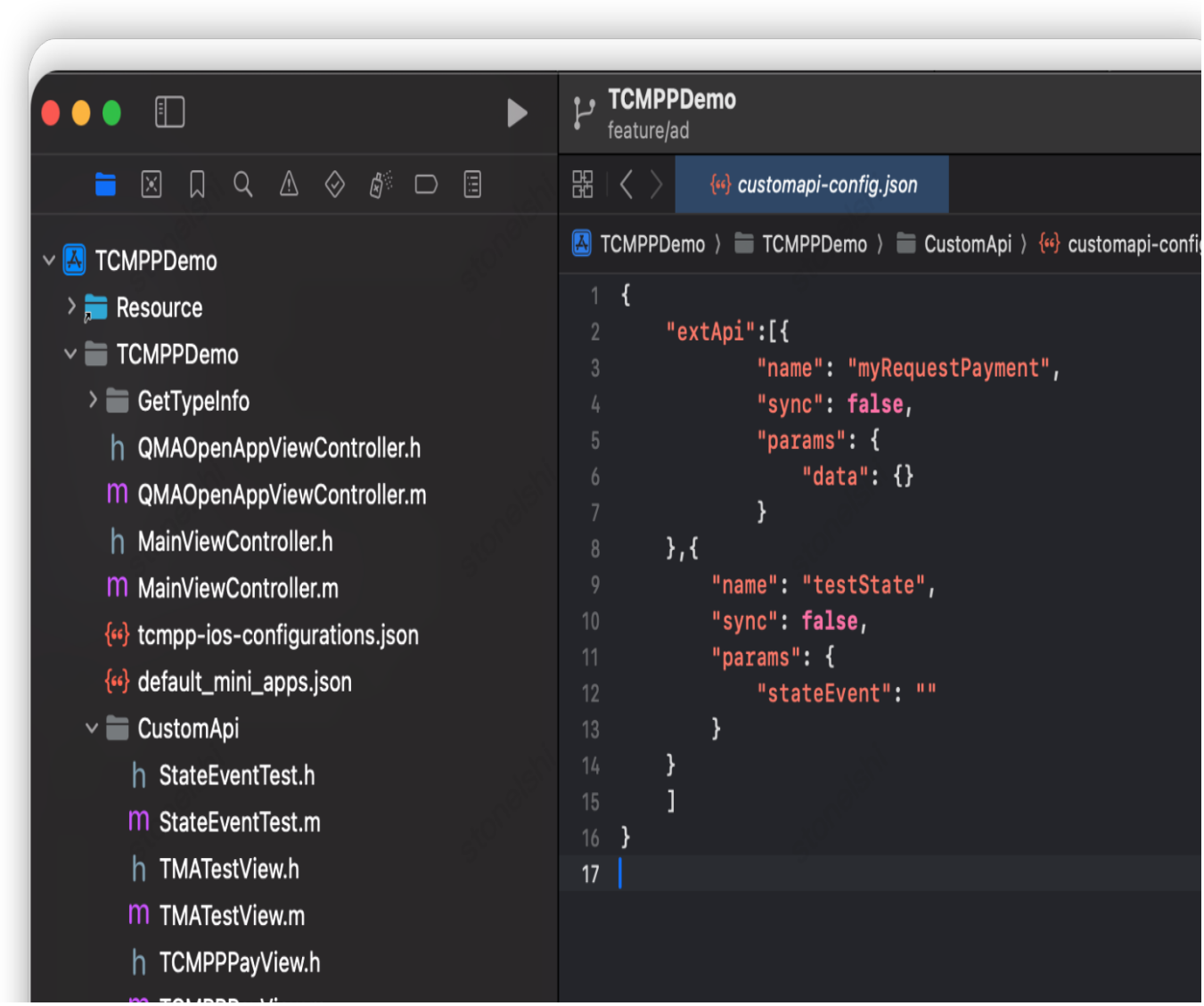
custom api supports configuration in the way of terminal app configuration file, which is invoked in the mini program by calling wx.api directly.

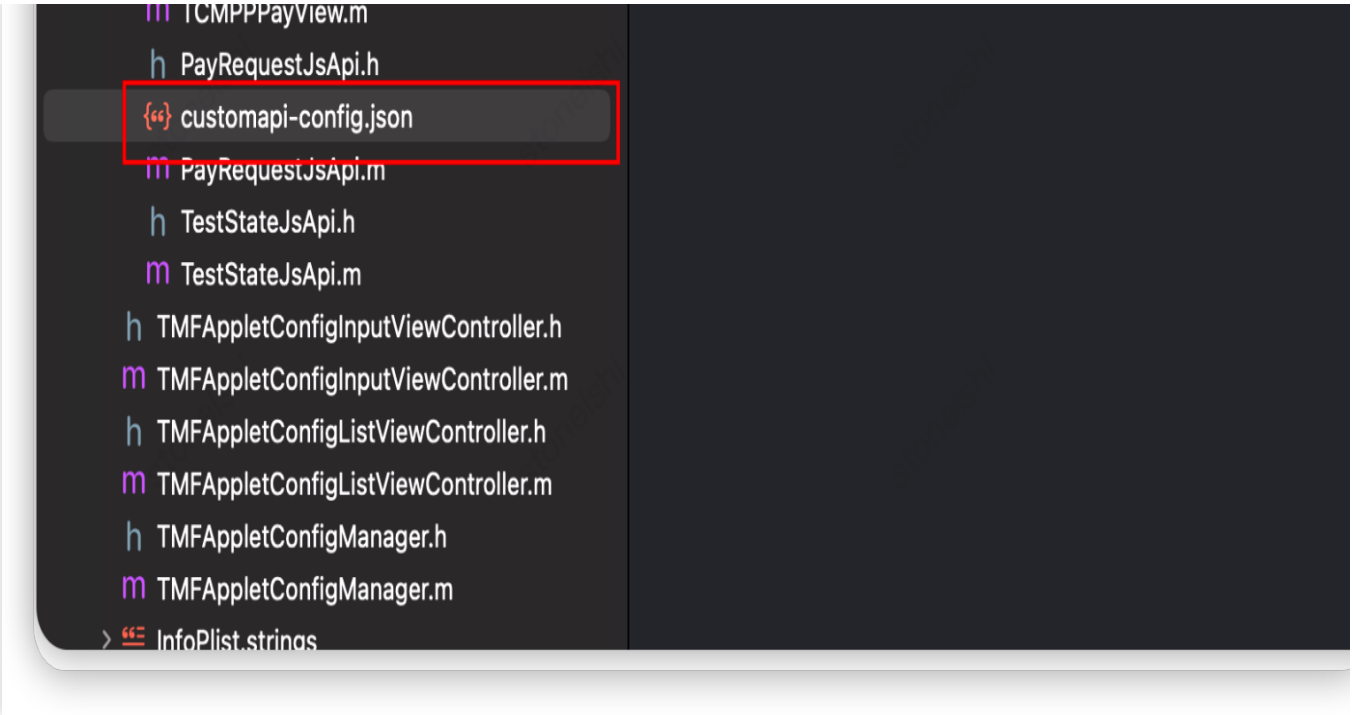
1. Unify the configuration file implemented in the app in customapi-config.json with the following reference:

```
{
    "extApi":[{
            "name": "test",
            "sync": false,
```

```
            "params": {
                "data": ""
            }
        },
        {
            "name": "testSync",
            "sync": true,
            "params": {
                "name": "",
                "title": ""
            }
        }
    ]
}
```

2. Put customapi-config.json into the iOS project:





3. Set the configuration file path during SDK initialization.

```
    [[TMFMiniAppSDKManager sharedInstance] setCustomApiConfigFile:[[NSBundle mainBun
```

4. Call it directly in the mini program with wx.test().

```
//Asynchronous API calls
var opts = {
    success: function(res) {},
    fail: function(res) {},
    complete: function(res) {},
    data: {
      name : 'kka',
      age : 22
    }
}
wx.test(opts);

//Synchronise api calls
var rst = testSync(opts);
```

Customize Mini Program View Components

Last updated：2025-01-16 19:14:27

Using customized native components

We support customized UI components. The customized component on the mini program side is <external-element>.

To use customized UI components, follow these steps:

1. Create a new class that inherits from UIView after integrating the SDK, e.g., QMATestView, and import the TMAExternalJSPlugin.h file. Ensure QMATestView conforms to the TMAExternalElementView protocol.
2. Register the QMATestView class as maTestView using the TMARegisterExternalElement method.
3. Implement the createWithParams and operateWithParams methods from the TMAExternalElementView protocol.

```
#import "QMATestView.h"
#import "TMAExternalJSPlugin.h"

@interface QMATestView () <TMAExternalElementView>

@end

@implementation QMATestView {
    UILabel *_textLabel;
    UIButton *_clickButton;
    id<TMAExternalJSContextProtocol> _context;
}

TMARegisterExternalElement(maTestView);
+ (UIView *)createWithParams:(NSDictionary *)params context:(id<TMAExternalJSContex
    QMATestView *testView = [[QMATestView alloc] initWithFrame:CGRectZero];
    NSDictionary *testViewParams = QQ_Dict_DictValue(params, @"params");
    [testView setText:QQ_Dict_StringValue(testViewParams, @"text")];
    testView->_context = context;
    return testView;
}

//Handle events called from the mini program side
- (void)operateWithParams:(NSDictionary *)param context:(id<TMAExternalJSContextPro
    NSDictionary *data = QQ_Dict_DictValue(param, @"data");
    NSDictionary *params1 = QQ_Dict_DictValue(data, @"params1");
    NSInteger age = [QQ_Dict_NumberValue(params1, @"age") integerValue];
    NSString *name = QQ_Dict_StringValue(params1, @"name");
    qq_weakify(self);
    [MAUtils executeOnThread:[NSThread mainThread] block:^{
      qq_strongify(self);
      if (self) {
```

```
        self->_textLabel.text = [NSString stringWithFormat:@"name = %@ , age = %ld"
        // Return the result to the mini program side
        TMAExternalJSPluginResult *result = [TMAExternalJSPluginResult new];
        result.result = @{@"result":@"success"};
        [context doCallback:result];
      }
    }];
}
```

Sending events to the mini program side

To send events to the mini program side from a customized native component, first record the context in the `createWithParams:context:` method:

```
_context = context;
```

Send an event as follows:

```
- (void)onClickButton:(UIButton *)button {
    _textLabel.text = @"What do you want me to do";
   //Assemble data and send the event
    NSString *data = [MAUtils JSONStringify:@{@"externalElementId":_elementId,@"typ
    [_context doSubscribe:kTMAOnExternalElementEvent data:data];
}
```

Usage on the mini program side

1. Include the following in the mini program wxml:

```
<external-element
    id="comp1"
    type="maTestView"
    _insert2WebLayer
    style="width: 200px;height: 100px;"
    bindexternalelementevent="handleEvent"
></external-element>
```

Note：

The `type` must be agreed upon with the native side. If not on the same layer, do not set the _insert2WebLayer attribute.

`bindexternalelementevent` can listen to operations passed from the native side, with callback parameters including:

```
{
target,
currentTarget,
```

```
timeStamp,
touches,
detail, // Parameters passed from native side
}
```

2. Create an instance in the mini program js.

```
this.ctx = wx.createExternalElementContext('comp1');
```

Note：

The parameter for wx.createExternalElementContext is the element id in wxml.

3. Call instance methods in the mini program:

```
this.ctx.call({
params1: {
name: 'name1',
age: 11
},
params2: {
name: 'name2',
age: 22
},
success: (e) => {
console.log('====operate success=====', e)
},
fail: (e) => {
console.log('====operate fail=====', e)
},
complete: (e) => {
console.log('====operate complete=====', e)
}
})
```

Note：

The instance provides a call method with success, fail, and complete callbacks. The base library will call the invoke method to pass parameters to the native side.

customize SDK Capabilities Introduction

Last updated：2025-06-06 19:02:51

To enhance usage scenarios, the mini program SDK has made some APIs available for customization by the superapp, allowing for unique capabilities.

Prerequisites

To implement custom APIs for the mini program SDK, you need to create a class that conforms to the TMFMiniAppSDKDelegate protocol:

```
#import <TCMPPSDK/TCMPPSDK.h>

@interface MIniAppDemoSDKDelegateImpl : NSObject <TMFMiniAppSDKDelegate>

@end
```

Once the prerequisites are met, you can proceed to customize the mini program SDK APIs according to the relevant documentation.

Documentation

Customize Mini Program UI
Customize Mini Program Information API
Customize Sharing Capability
Customize User Attributes
Logging and Event Reporting
Create Shortcut
Custom Player
Open APIs
Others

Customize Mini Program UI

Last updated：2025-05-26 15:24:41

The mini program SDK now allows developers to customize certain native UI effects of mini programs. Follow the guidelines below to configure this feature.

Set the loading page UI

The mini program engine enables you to customize the loading page of the mini program in the host app to replace the default loading page. This can be achieved by implementing the customLoadingViewWithAppInfo method in the TMFMiniAppSDKDelegate protocol. Here is an example:

```
- (UIView *)customLoadingViewWithAppInfo:(TMFMiniAppInfo *)appInfo frame:(CGRect)fr
    UIView *view = [[UIView alloc] initWithFrame:frame];
    // Set up the specific view content here

    return view;
}
```

Set mini program navigation bar resources

The mini program engine enables you to customize the mini program navigation bar resources in the host app to replace the default resources. This can be achieved by implementing the stringWithConfigKey method in the TMFMiniAppSDKDelegate protocol. The following keys are currently supported:

Key	Description
TMA_SK_MINIAPP_CloseButton	Close button icon.
TMA_SK_MINIAPP_CloseButtonDark	Close button in dark mode.
TMA_SK_MINIAPP_HomeButton	Home button icon.
TMA_SK_MINIAPP_HomeButtonDark	Home button icon in dark mode.
TMA_SK_MINIAPP_BackButton	Back button icon.
TMA_SK_MINIAPP_BackButtonDark	Back button icon in dark mode.
TMA_SK_MINIAPP_MoreButton	More button icon.

TMA_SK_MINIAPP_MoreButtonDark	More button icon in dark mode.
TMA_SK_MINIAPP_RecordButton	Record button icon.
TMA_SK_MINIAPP_RecordButtonDark	Record button icon in dark mode.
TMA_SK_MINIAPP_MoreBackground	Capsule button background.
TMA_SK_MINIAPP_MoreBackgroundDark	Capsule button background in dark mode.

Example:

```
- (NSString *)stringWithConfigKey:(NSString *)key {
    // Set the Close button in light mode
    if([key isEqualToString:TMA_SK_MINIAPP_CloseButton]) {
        return [[[NSBundle mainBundle] resourcePath] stringByAppendingPathComponent
    } else if([key isEqualToString:TMA_SK_MINIAPP_CloseButtonDark]) {
        return [[[NSBundle mainBundle] resourcePath] stringByAppendingPathComponent
    }


    return nil;
}
```

Set the mini program transition animations

You can customize the transition animations for mini program startup by implementing the getTMFSlideAnimationType method in the TMFMiniAppSDKDelegate protocol. The currently supported animations include: Bottom to top, Top to bottom, Left to right, Right to left, Default (Bottom to bottom)

```
// Set the mini program startup transition animation to Bottom to top
- (TMFSlideAnimationType)getTMFSlideAnimationType{
    return TMFSlideAnimationTypeBottomToTop;
}
```

Set mini program permission dialog

You can customize the authorization pop-up window of the mini program by implementing the createAuthorizeAlertViewWithFrame method in the TMFMiniAppSDKDelegate protocol. This method allows you to create a custom view for the authorization dialog, including the mini program and the permissions it needs to request.

```
/**
 * @brief - Create a custom authorization window
 *
 * @param frame - Window size
 * @param scope - The scope of the authorization, similar to WeChat authorization s
 * @param title - Permission name
 * @param desc - Permission description
 * @param privacyApi - The API being called that requires the permission
 * @param appInfo - Information of the current mini program
 * @param allowBlock - Callback for allowing the permission
 * @param denyBlock - Callback for denying the permission
 */

- (UIView *)createAuthorizeAlertViewWithFrame:(CGRect)frame
                                        scope:(NSString *)scope
                                        title:(NSString *)title
                                        desc:(NSString *)desc
                                        privacyApi:(NSString *)privacyApi
                                       appInfo:(TMFMiniAppInfo *_Nullable)appInfo
                                        allowBlock:(void (^)(void))allowBlock
                                         denyBlock:(void (^)(void))denyBlock;
```

Customize internal mini program UIs

You can customize the internal mini program UIs by implementing the corresponding methods in TMFMiniAppSDKDelegate. For details, see the following table:

Mini program API	TMFMiniAppSDKDelegate method
wx.showLoading	`- (void)showLoading:(TMALoadingInfo * _Nullable)infos;`
wx.hideLoading	`- (void)hideLoading;`
wx.showToast	`- (void)showToast:(TMAToastInfo *)infos;`
wx.hideToast	`- (void)hideToast;`
wx.showActionSheet	

(actionSheetType = 0)	`- (void)showActionSheetWithTitle:(nullable NSString *)title cancelButtonTitle:(nullable NSString *)cancelBut`
wx.showActionSheet (actionSheetType = 1)	`- (void)showShareViewWithTitle:(nullable NSString *)title cance`
wx.showModal	`- (void)showAlertWithTitle:(nullable NSString *)title withMessa`

Customize capsule button feature

Customize the close event listener for the Close button

You can customize the Close button event listener to allow the host app to receive callback events when the Close button is tapped.
Close button UI:

API description:

```
#pragma mark - Exit retention
- (BOOL)shouldDetainUser:(TMFMiniAppInfo *)app;
```

This API is triggered when the user taps the Close button. If this method returns YES, a dialog will appear to retain the user. If it returns NO, the mini program will exit directly.

Customize the more event listener for the More button

You can customize the More button event listener to allow the host app to receive callback events when the More button is tapped.
More button UI:

API description:

```
// Tap the Capsule button to call up the panel
/// If this method is not implemented, showActionSheetWithTitle:cancelButtonTitle:c
// @param app - Mini program information
// @param cancelButtonTitle - Cancel the title
// @param cancelAction - Cancel the operation
// @param otherButtonTitleAndActions - Other buttons and response operations
```

```
// @param dismissBlock - The operation that needs to be performed after the panel i
// @param parentVC - Call up the ViewController of the panel

- (void)showMoreButtonActionSheetWithApp:(TMFMiniAppInfo *)app
                         cancelButtonTitle:(nullable NSString *)cancelButtonTitle
                              cancelAction:(nullable dispatch_block_t)cancelActio
                 otherButtonTitleAndActions:(nullable NSArray *)otherButtonTitleAn
                              dismissBlock:(nullable dispatch_block_t)dismissBloc
                              parentVC:(UIViewController *)parentVC;
```

Customize the extended menu of the More button

When the user triggers the tap event for the More button, an extended menu appears to show some buttons. The default extended menu is shown as below:

Method 1

By overriding the customizedConfigForShare method, you can customize the sharing options and determine their display order.
API description:

```
// In the host app, you can customize the sharing channel and determine the display
// 1. Default channels: QQ, Qzone, WeChat, and Moments (for specific types, see MAU
// 2. Custom channels: Customized in the host app (type set to MAUIDelegateShareVie
// 3. Custom event processing: Host app customization (type set to MAUIDelegateShar
// The display order of the above three channels supports mixed arrangement.
///




- (NSArray<TMASheetItemInfo *> *)customizedConfigForShare;
```

Example:

```
- (NSArray<TMASheetItemInfo *> *)customizedConfigForShare {
    NSMutableArray *arrays = [[NSMutableArray alloc] init];
    TMASheetItemInfo *item1 = [[TMASheetItemInfo alloc] initWithTitle:@"More sharin
    item1.icon = [UIImage imageNamed:@"icon_moreOperation_shareChat"];
    [arrays addObject:item1];

    TMASheetItemInfo *item2 = [[TMASheetItemInfo alloc] initWithTitle:@"click" type
        NSLog(@"Tap");
```

```
    }];
    item2.icon = [UIImage imageNamed:@"icon_moreOperation_collect"];
    [arrays addObject:item2];
    return arrays;

}
```

The display effect is as follows:

Method 2

You can add or delete buttons on the extended menu by using customizedConfigForMoreButtonActions in the TMFMiniAppSDKDelegate protocol.

```
- (void)customizedConfigForMoreButtonActions:(NSMutableArray *)moreButtonTitleAndAc
    /*
    // Add a custom share channel
    TMASheetItemInfo *item = [[TMASheetItemInfo alloc] initWithTitle:@"Share" type:
    item.icon = [UIImage imageNamed:@"icon_moreOperation_shareChat"];
    [moreButtonTitleAndActions addObject:item];
     */

    /*
    // Delete the Copy link button
    NSMutableArray *newArrays = [[NSMutableArray alloc] initWithCapacity:moreButton
    for (TMASheetItemInfo *item in moreButtonTitleAndActions) {
        if(item.type != MAUIDelegateShareViewTypeCopyLink) {
            [newArrays addObject:item];
        }
    }

    [moreButtonTitleAndActions removeAllObjects];
    [moreButtonTitleAndActions addObjectsFromArray:newArrays];
     */
}
```

Customize image selection

The mini program SDK provides a default image selection feature, which uses the Android system's built-in gallery picker when the Mini Program calls the multimedia selection (wx.chooseImage).
You can customize the image selection by overriding the selectMediaFromPickerWithModel method.

```
// Select media from the image picker
// @param model - Configuration
// @param parentVC - ViewController
@param completionBlock - After the selection is completed, data needs to be returne
- (void)selectMediaFromPickerWithModel:(TMAMediaChooseConfigModel *)model
                             parentVC:(UIViewController *)parentVC
                      completionBlock:(void(^)(NSArray * _Nullable medias, NSError


// Shooting media
// @param model - Configuration
// @param parentVC - ViewController
@param completionBlock - After the selection is completed, data needs to be returne
- (void)selectMediaFromCameraWithModel:(TMAMediaChooseConfigModel *)model
                             parentVC:(UIViewController *)parentVC
                      completionBlock:(void(^)(id _Nullable media, NSError * _Null


// Get image data from PHAsset
// @param phAsset - Media object
// @param needCompress - Whether compression is required
- (NSData *)imageDataFromPhAsset:(PHAsset *)phAsset needCompress:(BOOL)needCompress
```

Customize image preview

The mini program SDK provides a default image preview implementation, which is triggered when the mini program calls the multimedia selection (wx.previewImage).
You can customize the image preview by overriding the - (void)navigationController:(UINavigationController *)navigationController presentImageWithCurrentUrl:(NSString *)currentAbsoluteUrl imageUrlArray:(NSArray*) absUrlsInPreviewArray method.
The default image preview page is shown as follows:

```
// Image preview
// @param navigationController - The navigation controller that calls up the image
// @param currentAbsoluteUrl - Current page address
// @param absUrlsInPreviewArray - Images to be previewed
- (void)navigationController:(UINavigationController *)navigationController
  presentImageWithCurrentUrl:(NSString *)currentAbsoluteUrl
               imageUrlArray:(NSArray *)absUrlsInPreviewArray;
```

Customize document preview

The iOS SDK provides a default document preview feature. You can customize the document preview feature by overriding the following API:

```
// Open document preview
// @param filePath - Path to the document
// @param titleName - Title
// @param appInfo appinfo - Mini program app info
- (void)openDocument:(NSString *)filePath
               title:(NSString *)titleName
             appInfo:(TMFMiniAppInfo *_Nonnull)appInfo;
```

Customize the web-view component loading progress bar

The mini program engine enables you to customize the loading progress bar for the web-view component in the host app by implementing the stringWithConfigKey in the TMFMiniAppSDKDelegate protocol.

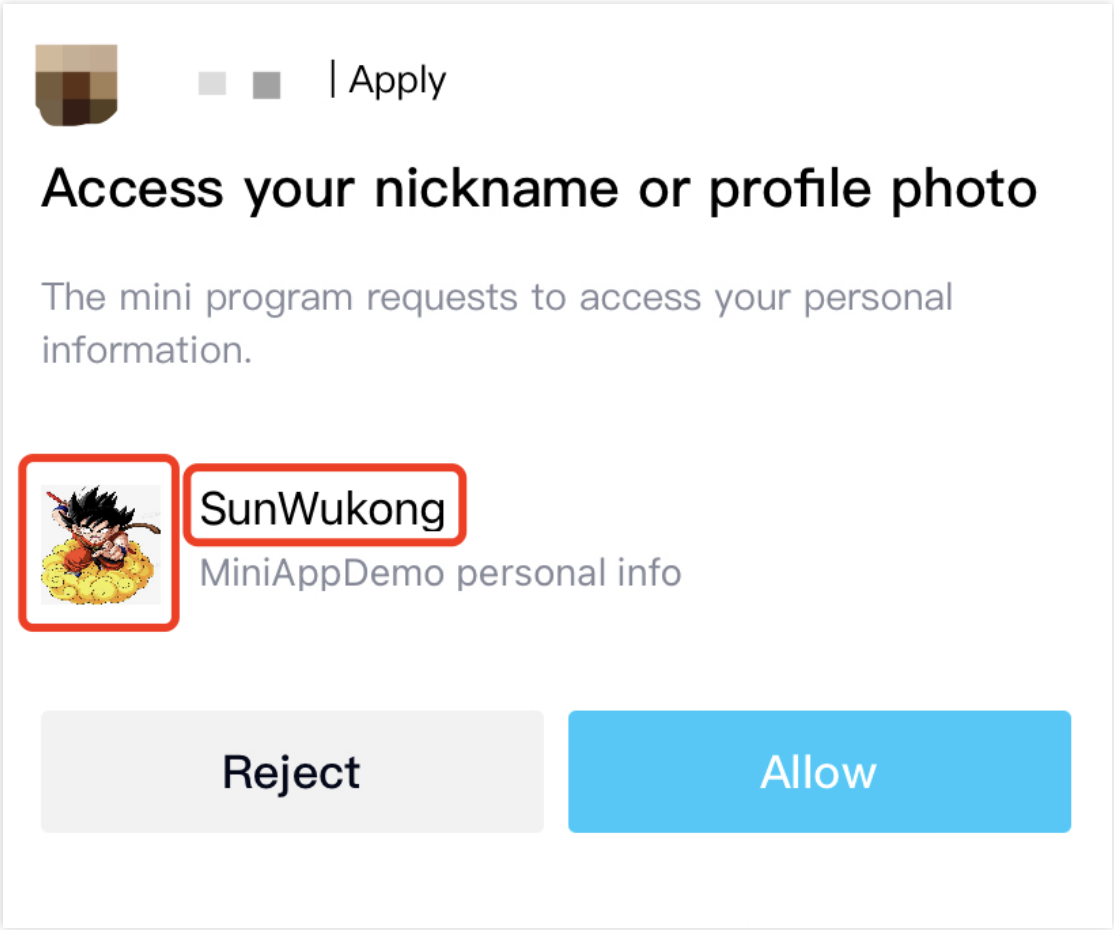
```
// You can configure whether to display the progress bar when the web-view componen

OBJC_EXTERN NSString* const TMA_SK_MINIAPP_WebView_Progress_Hide;

// You can configure the color of the progress bar when the web-view component load

OBJC_EXTERN NSString* const TMA_SK_MINIAPP_WebView_Progress_ProgressTintColor;

// You can configure the background color of the unfinished part of the progress ba

OBJC_EXTERN NSString* const TMA_SK_MINIAPP_WebView_Progress_TrackTintColor;


- (NSString *)stringWithConfigKey:(NSString *)key {
    if([key isEqualToString:TMA_SK_MINIAPP_WebView_Progress_Hide]) {
        return @"1";
    }
    return nil;
}
```

Customize Mini Program Information API

Last updated：2024-11-21 18:00:26

Customizing user information in authorization pop-up box

When a mini program requests user information, an authorization pop-up will appear. The host app can customize certain parts of this pop-up, including the profile image and username, as shown in the image below:



You can achieve this customization by overriding the fetchAppUserInfoWithScope method.

```
/**
 * @brief Fetch the profile photo and username from the SDK host platform based on
 - Pull the user avatar and nickname of the SDK host platform according to the scop
 */
- (void)fetchAppUserInfoWithScope:(NSString *)scope block:(TMAAppFetchUserInfoBlock
```

Sample code:

```
- (void)fetchAppUserInfoWithScope:(NSString *)scope block:(TMAAppFetchUserInfoBlock
    if (block) {
        UIImage *defaultAvatar = [UIImage imageWithContentsOfFile:[[[NSBundle mainB
        UIImageView *avatarView = [[UIImageView alloc] initWithImage:defaultAvatar]
        TMAAppUserInfo *userInfo = [TMAAppUserInfo new];
        userInfo.avatarView = avatarView;
        userInfo.nickName = @"SunWukong";
        block(userInfo);
    }
}
```

Customizing the userAgent in mini program WebView

Override the customUserAgent method to customize the user agent. Example code:

```
// Customized UA
// ua used by default
- (NSString *)customUserAgent:(NSString *)defaultUserAgent;
```

Customizing code scaning capability

The mini program SDK provides a default scan implementation (refer to the extension library support - Scan), and also offers an API for users to customize the scan capability.

```
// Scan the code to call the client's scan code module scancode - Scan the code to
// @param scanPrams Scan code parameter dictionary - scan code parameter dictionary
// @param navigationController Calls vc from the page - calls vc from that page
// @param completionHandler Callback result - callback result
- (void)scanCode:(NSDictionary *)scanPrams
navigationController:(UINavigationController *)navigationController
completionHandler:(MACommonCallback)completionHandler;
```

Monitoring mini program lifecycle

Override the lifeCycleForApp method to monitor the mini program lifecycle.

```
// Transparently transmit the application life cycle processing, and the business w
// @param appInfo Mini program information - The mini program information
```

```
// @param type Status - The state
- (void)lifeCycleForApp:(TMFMiniAppInfo *)appInfo type:(TMAAppLifeCycleType)type;
```

Customizing base library update strategy and monitoring

Override the canUpdateJSBaseLib method to monitor the update information of the mini program's base library.

```
// Base library update control - Basic library update control
// @param libInfo Update information including version: version number, url: downlo
// @return Whether to agree to update - Whether agree to update
- (BOOL)canUpdateJSBaseLib:(NSDictionary *)libInfo;
```

Customize Sharing Capability

Last updated：2024-11-21 18:00:49

Default Sharing Channels Configuration

The default sharing channels can be customised by overriding the defaultSharingChannels method.

API Description:

```
// Default sharing channel after clicking capsule button - MAShareTargetQ, MAShareT
// The default sharing channel after clicking the pill button - MAShareTarget  (MASh
// (The configuration of the mini program must be a subset of the App)
- (NSArray<NSNumber *> *)defaultSharingChannels;
```

Sample code:

```
- (NSArray<NSNumber *> *)defaultSharingChannels{
    return @[@(MAShareTargetQQ),@(MAShareTargetWXFriends),@(MAShareTargetWXMoment)]
}
```

Adding customised share buttons

By overriding the customisedConfigForShare method, you can customize the sharing route and determine the display order.

API description

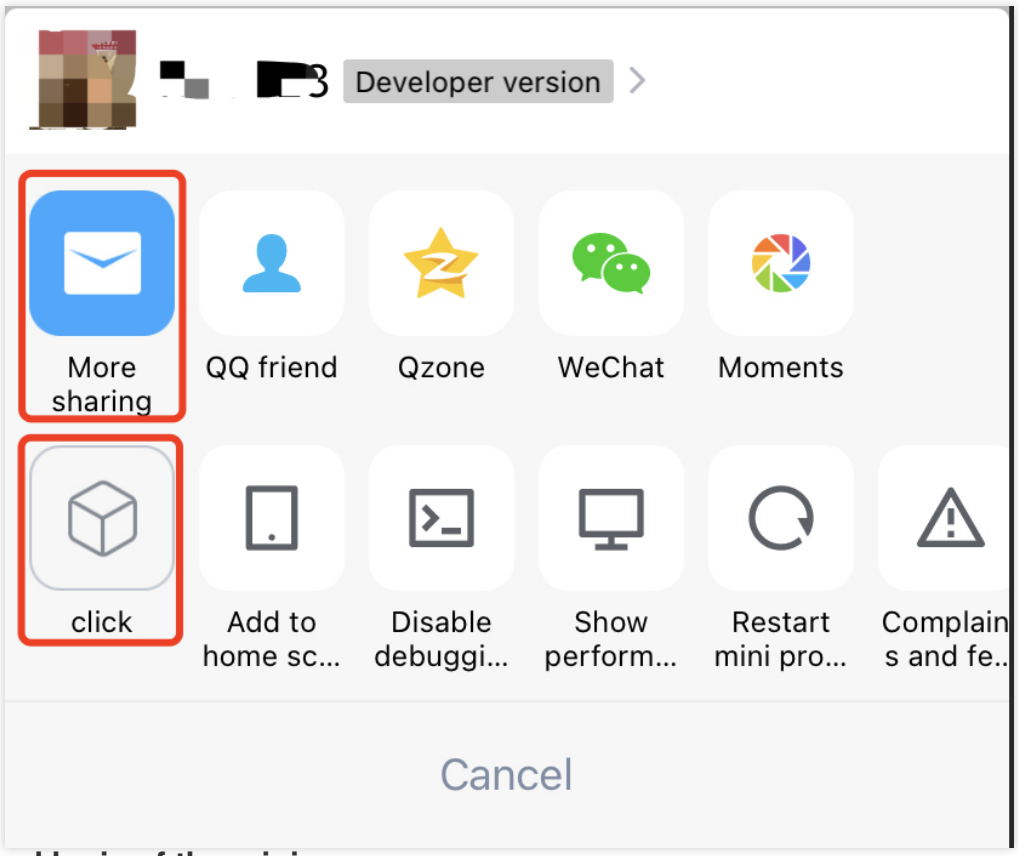
```
// The host App can customise the sharing channels and determine the display order,
// 1. Default channels: QQ friends, Qzone, WeChat, Circle of Friends (see MAUIDeleg
// 2, custom sharing channels: host App custom (type fill MAUIDelegateShareViewType
// 3, custom event processing: host App custom (type fill MAUIDelegateShareViewType
// The above three channels display order support mixed arrangement
///
// The host app can customise the share path and determine the display order. Curre
// 1. Default channels: QQ Friends, Qzone, WeChat, Moment (see MAUIDelegateShareVie
// 2. Custom sharing channels: host App custom (type fill in MAUIDelegateShareViewT
// 3. Custom event handling: host app custom (type fill in MAUIDelegateShareViewTyp
// The order in which the above three channels are displayed
- (NSArray<TMASheetItemInfo *> *)customisedConfigForShare;
```

Sample code:

```
- (NSArray<TMASheetItemInfo *> *)customizedConfigForShare {
    NSMutableArray *arrays = [[NSMutableArray alloc] init];
    TMASheetItemInfo *item1 = [[TMASheetItemInfo alloc] initWithTitle:@"More sharin
```

```
    item1.icon = [UIImage imageNamed:@"icon_moreOperation_shareChat"];
    item1.shareTarget = 10001;
    [arrays addObject:item1];

    TMASheetItemInfo *item2 = [[TMASheetItemInfo alloc] initWithTitle:@"click" type
        NSLog(@"click Sample code:");
    }];
    item2.icon = [UIImage imageNamed:@"icon_moreOperation_collect"];
    [arrays addObject:item2];
    return arrays;

}
```

The effect is as follows:



Internal logic of the mini program

The mini program internally listens for user clicks on the share class button via onShareAppMessage and customises the forwarded content.

Share Logic Implementation

After the host APP receives the sharing data returned from the mini program, it triggers shareMessageWithModel to execute the final sharing action, and the developer can process the content to be shared according to the shareTarget and other data by themselves, and docking to the three-party sharing platform.

```
/**
 * @brief shareModel Shared Interface - Shared Interface
 * @param shareModel shareModel - shared model
 * @param appInfo mini program appinfo - Mini program appinfo
 * @param completionBlock Callback - Callbacks
 */
- (void)shareMessageWithModel:(TMAShareModel *_Nonnull)shareModel
                      appInfo:(TMFMiniAppInfo *_Nonnull)appInfo
              completionBlock:(nullable void(^)(NSError * _Nullable error))completi
```

Customize Permission List

Last updated：2025-06-12 11:43:48

Certain APIs involve user privacy, and the Super App as a Service (SAS) enforces authorization controls for them. Users can choose whether to grant permissions based on their preferences. This doc covers customization requirements for the authorization list.

Get the mini program permission list

Call [TMFMiniAppSDKManager sharedInstance] getCurrentAppAuthorizeList to get the permission list for the specified mini program.

```
// Get the permission list of the currently running mini program

// @return NSArray<TMASettingItem *> *
- (NSArray *)getCurrentAppAuthorizeList;
```

Set mini program authorization status

Call [TMFMiniAppSDKManager sharedInstance] setCurrentAppAuthStatus: forScope: completionHandler to set the authorization status for the specified mini program.

```
// Change the authorization status of the currently running mini program

//@param status - Authorization status
//@param scope - Scope name
// @param completionHandler - Request result callback
- (void)setCurrentAppAuthStatus:(BOOL)status
                       forScope:(NSString *)scope
              completionHandler:(MACommonCallback _Nullable)completionHandler;
```

Customize mini program permission description

Permission list information can be customized by implementing the customizedScopeModelInfo API.

```
/**
 * @brief - Change the default scope description
```

```
 * @param appInfo - The current mini program information
 */
- (void)customizedScopeModelInfo:(NSArray<TMAScopeModel *> *)scopeModelList appInfo
```

For mini program permission list, refer to Scope list

Example:

```
- (void)customizedScopeModelInfo:(NSArray<TMAScopeModel *> *)scopeModelList appInfo
    for (TMAScopeModel *model in scopeModelList) {
        if(model.scope == ScopeChooseImage) {
            model.title = @"Choose Image";
            model.desc = @"Choose your image";
            model.settingPageTitle = @"Image";
        }
    }
}
```

Customize User Attributes

Last updated：2024-11-21 18:01:45

The host application developer can set the user's nickname and avatar information through the method provided by the SDK。

Setting up user nicknames

```
/**
 * @brief SDK host platform's user nickname, returns "TCMPP" by default -- SDK host
 */
- (NSString *)getAppNickName;
```

Setting up user avatars

```
/**
* @ brief SDK host platform user avatar, the default return demo image, SDK host pl
 */
- (UIImage *)getAppAvatar;
```

Get current user account identifier

```
/**
 * @brief Get the current user account identifier of the SDK host platform, usually

 * Note: Returning nil will invalidate some caches within the SDK. If you are not l
 */
- (NSString *_Nonnull)getAppUID;
```

Logging and Event Reporting

Last updated：2025-03-17 15:45:40

SDK runtime log printing

Implement the log output API during the development phase to facilitate troubleshooting.

Note：

It is recommended to disable log output when the app is released to ensure security and performance.

```
/// Print log
/// @param level Log level. For more information, see `PLTLogLevel`.
/// @param msg Log message
- (void)log:(MALogLevel)level msg:(NSString *)msg;
```

Reporting the event

The host app can implement the event reporting API to override the internal reporting logic of the SDK.

This includes mini program operation events and data reported by the mini program's internal wx.reportEvent.

```
typedef NS_ENUM(NSInteger,TMAReportEventID){.
    // undefined, report user-defined events
    // undefined, report user-defined event
    TMAReportEventID_None = 0,
    // Open the mini program.
    // Open the mini program.
    TMAReportEventID_OPEN_MINIAPP = 1,
    // update the mini program
    // Update the mini program.
    TMAReportEventID_UPDATE_MINIAPP = 2,
    // Download the mini program.
    // Download the mini program.
    TMAReportEventID_DOWNLOAD_MINIAPP = 3,
    // Open the mini program page.
    // Open the mini program page.
    TMAReportEventID_MINIAPP_PAGE_VIEW = 4,
    // Exit the mini program.
    // close the mini program
    TMAReportEventID_EXIT_MINIAPP = 5,
    // mini program behaviour event, atcion:0 background; 1 foreground
    // mini program behaviour event, atcion: 0 onHide; 1 onShow
    TMAReportEventID_MINIAPP_ACTION = 6
};

// Upload data - report data
// @param event event, refer to TMAReportEventID - event ID, refer to TMAReportEven
```

```
// @param eventName eventName - the name of the event
// @param params params - Parameters
// @param appInfo appletInfo - information about the mini program
// @return Whether to intercept internal reporting - Whether to intercept internal
- (BOOL)reportEvent:(int)eventId
          eventName:(NSString *)eventName
             params:(NSDictionary *)params
            appinfo:(TMFMiniAppInfo *)appInfo;
```

Real-time log reporting for mini programs

The host app can implement an event realtime log reporting interface to override the reporting logic inside the SDK. Including the log data written by calling wx.getRealtimeLogManager inside the mini program.

```
// Report log data - Report log data
// @param appId appID
// @param jsVersion base library version - base library version
// @param page current page - Current page
// @param filterMsgs Filtered Content - Filtered Content
// @param logs log events - log events
// @param completionBlock - The result of the callback.
// @return Whether to block internal reporting - Whether to block internal reportin
- (BOOL)reportRealTimeLogWithAppId:(NSString *)appId
                         jsVersion:(NSString *)jsVersion
                              page:(NSString *)page
                          filterMsgs:(NSArray <NSString *>*)filterMsgs
                              logs:(NSArray <TMARealtimeLogItem *>*)logs
                   completionBlock:(void (^)(NSError * _Nullable error))completionB
```

Internal log reporting for mini programs

The host app can implement event real-time log reporting interface to override the reporting logic inside the SDK. Including the log data written by wx.getLogManager within the mini program, and the user can upload the printed log by using the open-type="feedback" of the button component.// Upload the logs of the mini program corresponding to the appID - Upload the logs of the mini program corresponding to the appID

```
// Upload the logs of the mini program corresponding to the appID - Upload the logs
// Implemented using TMFMiniAppSDKManager's `sandBoxPathWithAppID:` interface to ge
// @param appID appID of the mini program - appID of the applet/game.

- (void)uploadLogFileWithAppID:(NSString *)appID;
```

Create Shortcut

Last updated：2025-05-26 15:26:42

How to create shortcut

Tap More "···" in the top-right corner, and find the Create shortcut in the bottom of the screen.

After tapping, it will redirect to the Safari page.

Tap Add to Home Screen to create a shortcut for the mini program.

Complete the configurations to use the feature

1. The client needs to return the scheme protocol name via the **getAppScheme** delegate method. Example code:

```
- (NSString *)getAppScheme{
    return @"xxx";
}
```

2. Configure the same scheme in the URL Types section of the main project's info.plist file.

3. Implement the handleOpenUrl method in the openUrl method. See the example below:

```
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url options:(NSDictionary
    if ([[TMFMiniAppSDKManager sharedInstance] handleOpenUrl:url]){
        return YES;
    }
    return YES;
 }
```

Once completing the above steps, you can open the released version of the mini program by scanning the console QR code with the system camera.

Custom Player

Last updated：2025-06-03 18:00:36

If the superapp has integrated third-party players or developed its own, it can use the delegate API to replace the mini program's default player with its existing player.
To implement a custom player, the prerequisite is to customize the mini program SDK APIs by creating a class that conforms to the TMFMiniAppSDKDelegate protocol:

```
#import <TCMPPSDK/TCMPPSDK.h>

@interface MIniAppDemoSDKDelegateImpl : NSObject <TMFMiniAppSDKDelegate>

@end
```

You need to implement the corresponding delegate method to return the player instance that conforms to the protocol when creating the player instance:

```
- (id<TMAVideoPlayerDelegate>)app:(TMFMiniAppInfo *_Nonnull)app createVideoViewWith
```

The TMAVideoPlayerDelegate API provides various properties (such as playback URL, mute status, and whether to show the progress bar) and events (such as play, pause, playback speed, and entering/exiting full screen) that can be implemented in the corresponding methods. The complete API is as follows:

```
@property (nonatomic, strong) NSString *tmaVideoPlayerCurrentURL; // Playback URL

/*
 tmaVideoPlayerHostVC is used to record the VC where the video player is located, u
 No other operations are required.


 */
@property(nonatomic, weak)   UIViewController<TMAVideoViewFullScreenProtocol> *tmaV

// Cover image URL

@property(nonatomic, strong) NSString *tmaVideoPlayerPosterUrl;

// Specify the initial playback position

@property(nonatomic, assign) CGFloat tmaVideoPlayerInitialTime;

// Whether to show control buttons
```

```
@property(nonatomic, assign) BOOL tmaVideoPlayerShowControls;

// Whether to repeat playback

@property(nonatomic, assign) BOOL tmaVideoPlayerRepeat;

// Whether to mute

@property(nonatomic, assign) BOOL tmaVideoPlayerIsMuted;

// Whether to show the center play button

@property(nonatomic, assign) BOOL tmaVideoPlayerShowCenterPlayButton;

// Control videoGravity, contentsGravity, and poster's contentMode with three value

@property(nonatomic, strong) NSString *tmaVideoPlayerObjectFitMode;

// Full screen orientation with three values: 0 UIInterfaceOrientationPortrait, 90

@property(nonatomic, strong) NSNumber *tmaVideoPlayerFullscreenDirection;

// Whether to support double-tap to pause/play

@property(nonatomic, assign) BOOL tmaVideoPlayerEnablePlayGesture;

// The title in the top left corner in full-screen mode

@property(nonatomic, strong) NSString *tmaVideoPlayerFullscreenTitle;

// Whether to show the progress bar

@property(nonatomic, assign) BOOL tmaVideoPlayerShowProgress;

// Whether to show the full screen button

@property(nonatomic, assign) BOOL tmaVideoPlayerShowFullscreenBtn;

// Whether to show the play button in the bottom control bar

@property(nonatomic, assign) BOOL tmaVideoPlayerShowPlayBtn;

// Whether to show the mute button

@property(nonatomic, assign) BOOL tmaVideoPlayerShowMuteButton;
```

```
// Whether to support progress adjustment gestures

@property(nonatomic, assign) BOOL tmaVideoPlayerEnableProgressGesture;

// Whether to enable volume and brightness adjustment gestures when not in full scr

@property(nonatomic, assign) BOOL tmaVideoPlayerPageGestureWhenNotFullscreen;
@property(nonatomic, strong) NSArray *tmaVideoPlayerDanmuList;

// Whether to autoplay

@property(nonatomic, assign) BOOL tmaVideoPlayerAutoPlay;

// Whether to allow background playback

@property(nonatomic, assign) BOOL tmaVideoPlayerBackgroundPlayback;

// Whether to enable picture-in-picture

@property(nonatomic, assign) BOOL tmaVideoPlayerIsPictureInPicture;

// Records whether it is in full-screen mode

@property(nonatomic, assign) BOOL tmaVideoPlayerIsFullScreen;

// Whether to automatically rotate when entering full screen

@property(nonatomic, assign) BOOL tmaVideoPlayerEnableAutoRotation;



/*
 Used to send player events to the mini program

 */
@property(nonatomic, strong) void(^ _Nullable tmaVideoPlayerStatusBlock)(MAVideoPla

// Play

- (void)tmaVideoPlayerPlay;

// Pause

- (void)tmaVideoPlayerPause;

// Stop
```

```
- (void)tmaVideoPlayerStop;

// Adjust the playback progress

- (void)tmaVideoPlayerSeekDuration:(CGFloat)duration;

// Playback rate

- (void)tmaVideoPlayerPlayBackRate:(CGFloat)rate;

// Enter full screen

- (void)tmaVideoPlayerEnterFullScreen;

// Exit full screen

- (void)tmaVideoPlayerExitFullScreen;

// Whether to show the system status bar in full screen

- (void)tmaVideoPlayerShowStatusBar:(BOOL)show;

// Insert on-screen comment

- (void)tmaVideoPlayerInsertDanmuText:(NSString *_Nonnull)content color:(NSString *

// Interrupt playback

- (void)tmaVideoPlayerInterruptPlay;

// Resume playback

- (void)tmaVideoPlayerResumePlay;

// Pause when the page disappears

- (void)tmaVideoPlayerPauseWhenDiappear;

// Play when the page resumes

- (void)tmaVideoPlayerPlayWhenAppear;

// Whether to support background playback

- (void)tmaVideoPlayerRequestBackgroundPlayback:(BOOL)state;
```

Open APIs

Last updated：2025-01-09 15:27:43

The mini program SDK provides some open APIs for implementing features that are provided by host apps, such as login, user information retrieval, and payment.

Note:

Starting from version 2.1.0, the SDK APIs come with default implementations for login (wx.login), user information (wx.getPhoneNumber, wx.getEmail, wx.chooseAvatar, wx.getNickName), and session status checks (wx.checkSession). This allows developers to easily configure and integrate user login and host application user information logic for mini programs.

The default implementation starting from version 2.1.0 is **only supported on SaaS**.

SDK version 2.1.0 and later

For SDK version 2.1.0 and later, the following table shows the custom open APIs:

Mini program method	TMFMiniAppSDKDelegate proxy implementation method	Description	Default implementation
wx.login	login	Login API SDK.	Default implementation from version 2.1.0
wx.getPhoneNumber	getPhoneNumber	Retrieves phone number.	Default implementation from version 2.1.0
wx.checkSession	checkSession	Checks if the login session is expired.	Default implementation from version 2.1.0
wx.getEmail	getEmail	Retrieves user email.	Default implementation from version 2.1.0
wx.chooseAvatar	chooseAvatar	Retrieves the user profile photo.	Default implementation from version 2.1.0
wx.getNickName	getNickName	Retrieves the user nickname.	Default implementation from version 2.1.0
wx.getUserInfo	getUserInfo	Retrieves basic user information.	No default implementation
wx.getUserProfile	getUserProfile	Retrieves user profile information.	No default implementation

wx.requestPayment	requestPayment	Initiates a payment.	No default implementation

App developers can override the TMFMiniAppSDKDelegate proxy methods to implement logic for retrieving basic user information (wx.getUserInfo), user profile information (wx.getUserProfile), and initiating payments (wx.requestPayment).
Starting from version 2.1.0, if you want to use the default implementation, you need to return the obtained UID (the current user account identifier on the host platform) to the SDK engine through the proxy API after logging into the host app. For example, after logging in to the host app, if the UID is set in [DemoUserInfo sharedInstance].userId, the implementation of the proxy API is as follows:

```
- (NSString *)getAppUID {
    return [DemoUserInfo sharedInstance].userId;
}
```

Related APIs in the proxy class

```
// Initiates a payment
// @param app mini program/mini game instance
// @param params parameters
// @param completionHandler callback results
- (void)requestPayment:(TMFMiniAppInfo *)app params:(NSDictionary *)params completi

// login
// @param app mini program/mini game instance
// @param params parameters
// @param completionHandler callback results
- (void)login:(TMFMiniAppInfo *)app params:(NSDictionary *)params completionHandler

// checkSession
// @param app mini program/mini game instance
// @param params parameters
// @param completionHandler callback results
- (void)checkSession:(TMFMiniAppInfo *)app params:(NSDictionary *)params completion

// getUserProfile
// @param app mini program/mini game instance
// @param params parameters
// @param completionHandler callback results
- (void)getUserProfile:(TMFMiniAppInfo *)app params:(NSDictionary *)params completi


// getPhoneNumber
```

```
// @param app mini program/mini game instance
// @param params parameters
// @param completionHandler callback results
- (void)getPhoneNumber:(TMFMiniAppInfo *)app params:(NSDictionary *)params completi

// getUserInfo
// @param app mini program/mini game instance
// @param params parameters
// @param completionHandler callback results
- (void)getUserInfo:(TMFMiniAppInfo *)app params:(NSDictionary *)params completionH

// Whether to use the custom OpenApi
// @param app mini program/mini game instance
- (BOOL)whetherToUseCustomOpenApi:(TMFMiniAppInfo *)app;
```

Example

```
- (void)getUserInfo:(TMFMiniAppInfo *)app params:(NSDictionary *)params completionH
    if (completionHandler) {
        completionHandler(@{
            @"nickName": [DemoUserInfo sharedInstance].nickName,
            @"avatarUrl": [DemoUserInfo sharedInstance].avatarUrl,
            @"gender": [NSNumber numberWithUnsignedInt:[DemoUserInfo sharedInstance
            @"country": [DemoUserInfo sharedInstance].country,
            @"province": [DemoUserInfo sharedInstance].province,
            @"city": [DemoUserInfo sharedInstance].city,
            @"language": @"zh_CN"
        },nil);
    }
}

- (void)getUserProfile:(TMFMiniAppInfo *)app params:(NSDictionary *)params completi
    if (completionHandler) {
        completionHandler(@{
            @"nickName": [DemoUserInfo sharedInstance].nickName,
            @"avatarUrl": [DemoUserInfo sharedInstance].avatarUrl,
            @"gender": [NSNumber numberWithUnsignedInt:[DemoUserInfo sharedInstance
            @"country": [DemoUserInfo sharedInstance].country,
            @"province": [DemoUserInfo sharedInstance].province,
            @"city": [DemoUserInfo sharedInstance].city,
            @"language": @"zh_CN"
        },nil);
    }
}
```

SDK version 2.0.15 and earlier

For SDK version 2.0.15 and earlier, the following table shows the custom open APIs:

Mini program method	TMFMiniAppSDKDelegate method	Description
wx.login	login	Login API.
wx.getUserInfo	getUserInfo	Retrieves basic user information.
wx.getUserProfile	getUserProfile	Retrieves user profile information.
wx.getPhoneNumber	getPhoneNumber	Retrieves phone number.
wx.requestPayment	requestPayment	Initiates a payment.
wx.checkSession	checkSession	Checks if the login session is expired.

Users can implement the TMFMiniAppSDKDelegate proxy to link data interactions between the mini program and the host app.

Compatibility of custom login logic

Since SDK version 2.1.0 and later come with built-in login and user information retrieval logic, existing customers who have already implemented custom user login logic by overriding the TMFMiniAppSDKDelegate proxy need to configure the SDK to continue using their custom logic after updating to version 2.1.0 or later. Here's how to do it:

```
// Whether to use the custom OpenApi
// Returns YES to disable the default login implementation, NO to use the default l
// @param app mini program/mini game instance
- (BOOL)whetherToUseCustomOpenApi:(TMFMiniAppInfo *)app;
```

Note:

Mini program developers use the **apiVersion** field in app.json to control versions of the APIs starting from 2.1.0 version with default implementation. When the field is not configured, uses the value returned by the **whetherToUseCustomOpenApi** proxy API. When the field is configured, keep the API version for consistency. That means,
the priority of apiVersion configuration is higher than the value returned by the whetherToUseCustomOpenApi proxy API.

For example: If the apiVersion is configured to V2, all APIs will use the default implementation, even if whetherToUseCustomOpenApi returns YES.

Others

Last updated：2025-05-26 15:25:12

Host app APIs

```
/**
 * @brief SDK host app name.
 * This API is mainly used to indicate the app name.
 */
- (NSString *)appName;

/**
 * @brief SDK host app version.
 * @return Returns a lowercase string, such as 1.0.0, is returned.
 */
- (NSString *)getAppVersion;

/**
 * @brief The network status of the SDK host app.
 */

- (TMANetWorkStatus)getAppNetworkStatus;

/**
 * @brief Model information of the SDK host app.
 */
- (NSString *)getAppIPhoneModel;

/**
 * @brief Device information of the SDK host app.
 * @return format: {@"brand":@"iPhone",@"model":@"iPhone 11<iPhone12,1>",@"system":
 */
- (NSDictionary *)getAppDeviceInfo;
/**
 * @brief Obtains basic information of the host app.
 * @return format: {@"SDKVersion":@"2.32.2",@"model":@"iPhone 11<iPhone12,1>",@"sys
 */
- (NSDictionary *)getAppBaseInfo;

/**
 * @brief Obtains current language set by the host app.
 * @return format: "zh-Hans"
 */
- (NSString *)getCurrentLocalLanguage;
```

```
/**
 * @brief If the current theme set by the host is not implemented, the theme return
 */
- (NSString *)getAppTheme;

/**
 * @brief Clipboard frequency control.
 */
- (NSNumber *)getClipboardInterval

// The maximum number of active mini programs. Default value: 3.

- (NSInteger)maxMiniAppKeepAliveCount;

// Sets the URL Scheme to open the host app of the mini program.

- (NSString *)getAppScheme;
```

Screen capture, screen recording and watermarking.

```
- (void)applet:(TMFMiniAppInfo *)appletInfo screenCaptureStatusChanged:(BOOL)isCapt

- (void)appletDidTakeScreenshot:(TMFMiniAppInfo *)appletInfo atPagePath:(NSString *

- (nullable UIView *)appletCustomizeWatermarkView:(TMFMiniAppInfo *)appletInfo;
```

Processing special URLs in web-view component

Special URLs in web-view component pages should be passed to the host app for processing.

```
// Triggered when redirecting to non-http/https in the web-view component. It is us
// @param app {TMFMiniAppInfo} Information about the mini program where the web-vie
// @param url {NSURL} The URL to be accessed.
// @return Whether to intercept.
- (BOOL)webViewCustomUrlLoading:(TMFMiniAppInfo *)app url:(NSURL *)url {
    NSLog(@"webViewCustomUrlLoading:%@,appid:%@",[url absoluteString],app.appId);
    if ([[UIApplication sharedApplication] canOpenURL:url]) {
        if (@available(iOS 10.0, *)) {
            [[UIApplication sharedApplication] openURL:url options:@{} completionHa
                if (success) {
                    NSLog(@"webViewCustomUrlLoading:%@,appid:%@, open sucess",[url
                }
            }];
        } else {
            [[UIApplication sharedApplication] openURL:url];
```

```
        }
        return YES;
    } else {
        NSLog(@"webViewCustomUrlLoading:%@,appid:%@,cann't open!!!",[url absoluteSt
    }
    return NO;
}
```

Compatibility for HTTP resources on iOS 18 and above

Due to system security restrictions on iOS 18 and above, if you use HTTP resources in a mini program, you need to enable "App Transport Security Settings - Allow Arbitrary Loads" in your project. Implement the TMFMiniAppSDKDelegate protocol and use the stringWithConfigKey method to return the appropriate value for TMA_SK_MINIAPP_ATS_Allow_Arbitrary_Loads to enable this setting.

```
//TMFMiniAppSDKDelegate protocol
- (NSString *)stringWithConfigKey:(NSString *)key {
    if([key isEqualToString:TMA_SK_MINIAPP_ATS_Allow_Arbitrary_Loads]) {
        return @"1";
    }
    return nil;
}
```

API Introduction

Last updated：2024-11-21 18:04:43

TMAVersionType

Mini program version

```
    //Development version
    // Development version
    TMAVersionDevelop = 0,

    //Preview version
    // Preview version
    TMAVersionPreview = 1,

    //Reviewed version
    // Audit version
    TMAVersionAudit = 2,

    //Online version (released version)
    // Online version (release version)
    TMAVersionOnline = 3,

    //Local preloaded version, will not be updated (Not available currently)
    // Local preset version will not be updated (not used yet)
    TMAVersionLocal = 10,
```

TMAEntryScene

The scene where the mini program is opened.

```
    // Initialization value
    // initialization value
    TMAEntrySceneNone                                = 0,

    // Main entry
    // main entrance
    TMAEntrySceneAIOEntry                            = 1001,

    // Search
    // search
    TMAEntrySceneSearch                              = 1005,

    // Scan QR code
    //Scan QR code
```

```
    TMAEntrySceneScanQRCode                              = 1011,
```

MAShareTarget

```
    //Share to QQ
    // Share to QQ
    MAShareTargetQQ = 0,

    //Share to Qzone
    // Share to Qzone
    MAShareTargetQzone,

     | 2 | Share to QQ, quickly share to the current chat window |
    // Share to QQ, and share to the current chat window quickly
    MAShareTargetQQFastForward,

    //WeChat friends
    // WeChat friends
    MAShareTargetWXFriends,

    // WeChat Moments
    // WeChat Moments
    MAShareTargetWXMoment,

    //Quick sharing of recent contacts, triggered by clicking on the sharing panel
    // Quick sharing of recent contacts, triggered by clicking on the sharing
    MAShareTargetChatFastShare,

    // Share contacts quickly
    //Quickly share friends list
    MAShareTargetFastShareTable,

    //openId sharing, click and share the open data domain of the source mini game
    // openId sharing, click and share the open data domain of the source mini game
     MAShareTargetOpenIdShare,
    MAShareTargetGuild,
```

TMAAppLifeCycleType

```

     TMAAppLifeCycleTypeNone,

     TMAAppLifeCycleTypeonColdOpen,

     TMAAppLifeCycleTypeonHotOpen,
```

```
    TMAAppLifeCycleTypeOnShow,

    TMAAppLifeCycleTypeOnHide,

    TMAAppLifeCycleTypeOnClose,

    TMAAppLifeCycleTypeTerminate,
```

Extension SDK

Last updated：2025-06-12 15:56:36

Many APIs that require system authorization for development and use need to be pre-authorized in the info.plist file of the project. However, your app may not need this feature, so the SDK has been split into core and extension modules, eliminating unnecessary permissions and reducing the size of the core module.
The SDK engine provides both core and extension modules, allowing users to integrate based on their specific needs.

Integration and usage of the extension SDKs

The extension SDK complements the core SDK, so to use the extension SDK, you must also depend on the core SDK. This separation ensures the security and stability of the SDK by placing APIs that require permissions into the extension SDK.

TCMPPExtMedia

TCMPPExtMedia provides the default implementations of three APIs: chooseMedia, chooseVideo, and chooseImage. If the host app already has these capabilities, we recommend you implement them in open APIs. To use the media selection feature provided by TMF, you need to use this plugin.
How to use:

```
pod 'TCMPPExtMedia'
```

Add following lines to the info.plist file: `Privacy - Photo Library Usage Description`
After the Media extension SDK is added, the supported mini program APIs are as follows:

API name	Description
wx.chooseMedia	Selects an image or video.
wx.chooseVideo	Selects a video.
wx.chooseImage	Selects an image.

Required permissions:

Permission	Description
Gallery access	Required to access the gallery.

TCMPPExtScanCode

TCMPPExtScanCode module provides the processing logic for wx.scanCode. If the host app has code scanning and recognition capabilities, it is recommended to integrate them through:
`TMFMiniAppSDKDelegate.scanCode:(NSDictionary *)scanPrams navigationController:(UINavigationController *)navigationController completionHandler:(MACommonCallback)completionHandler;`
However, if you need to use the code scanning feature provided by the SDK, you need to use this plugin.
How to use:

```
pod 'TCMPPExtScanCode'
```

Add following lines to the info.plist file: `Privacy - Camera Usage Description`
After the QR code scanning extension SDK is added, the supported mini program APIs are as follows:

API name	Description
wx.scanCode	Opens the client's QR code scanning interface

Required permissions:

Permission	Description
Camera	Required for QR code scanning.
Gallery access	Required to access the gallery.

TCMPPExtLive

If you need to use the live streaming components (live-player and live-pusher) for developing capabilities of live streaming push and pull, you need to add the following SDKs.

```
pod "TCMPPExtLive"
pod 'TXLiteAVSDK_Professional', :podspec => 'https://liteav.sdk.qcloud.com/pod/lite
```

In addition to adding the above dependencies, you need to implement the following methods in TMFMiniAppSDKDelegate to provide the LicenseUrl and LicenseKey required for the live streaming component. This is necessary to complete the initialization configuration for the live streaming component. If you do not configure the correct LicenseUrl and LicenseKey, the live streaming component will not function properly.
Note:
For the method of obtaining the LicenseURL and LicenseKEY, see Adding and Renewing A License .

```
- (NSString *)setLiveLicenceURL {
    return @"https://xxx.license";
}

- (NSString *)setLiveLicenceKey {
    return @"xxx";
}
```

Add following lines to the info.plist file:

`Privacy - Camera Usage Description`

`Privacy - Microphone Usage Description`

After the live streaming extension SDK is added, the supported mini program APIs are as follows:

API name	Description
wx.createLivePusherContext	Creates the live streaming pusher context.
LivePusherContext	Supports LivePusherContext APIs.
wx.createLivePlayerContext	Creates a live streaming player context.
LivePlayerContext	Supports LivePlayerContext APIs.
Components	-
live-pusher	Tag for live streaming push.
<live-player>	Tag for live streaming play.

Required permissions:

Permission	Description
Camera permission	-
Recording permission	-

TCMPPExtAuthentication

TCMPPExtAuthentication module provides capabilities related to biometric authentication.

Integration method:

```
pod "TCMPPExtAuthentication"
```

How to use:

Add following lines to the info.plist file: `Privacy - Face ID Usage Description`

After the biometric authentication extension SDK is added, the supported mini program APIs are as follows:

API name	Description
wx.startSoterAuthentication	-
wx.checkIsSupportSoterAuthentication	-
wx.checkIsSoterEnrolledInDevice	-

Required permissions:

Permissions	Description
Biometric authentication access	Requires the biometric authentication permission.

TCMPPExtBLE

TCMPPExtBLE module provides Low Energy (BLE) Bluetooth and beacon related capabilities.

Integration method:

```
pod "TCMPPExtBLE"
```

How to use:

Add following lines to the info.plist file:

`Privacy - Bluetooth Always Usage Description`

`Privacy - Bluetooth Peripheral Usage Description`

After the LBS extension SDK is added, the supported mini program APIs are as follows:

APIs	Description
Bluetooth - general-purpose	General-purpose Bluetooth API.
Bluetooth Low Energy (BLE) peripheral device	Bluetooth Low Energy (BLE) peripheral devices APIs.
Bluetooth-Low Energy Central Device	Center device APIs.
Bluetooth - Beacon	Bluetooth beacon APIs.

Required permissions:

Permissions	Description
Bluetooth	Required to operate the Bluetooth.
Location	Required to search the Bluetooth device.

TCMPPExtCalendar

TCMPPExtCalendar module provides calendar-related capabilities.

Integration method:

```
pod "TCMPPExtCalendar"
```

How to use:

Add following lines to the info.plist file:

`Privacy - Calendars Usage Description`

`Privacy - Reminders Usage Description`

API list:

API name	API description
addPhoneRepeatCalendar	Adds a recurring event to the system calendar.
addPhoneCalendar	Adds an event to the system calendar.

TCMPPExtClipBoard

TCMPPExtClipBoard module provides clipboard-related capabilities.

Integration method:

```
pod "TCMPPExtClipBoard"
```

API list:

API name	API description
setClipboardData	Sets the content of the system clipboard.

getClipboardData	Gets the content of the system clipboard.

TCMPPExtContact

TCMPPExtContact provides contact-related capabilities.

Integration method:

```
pod "TCMPPExtContact"
```

How to use:

Add following lines to the info.plist file: `Privacy - Contacts Usage Description`

API list:

API name	API description
chooseContact	Selects a contact.
addPhoneContact	Adds a contact to the phone's contact list.

TCMPPExtLBS

TCMPPExtLBS module provides system positioning, system map, compass, accelerometer, device motion, and gyroscope related capabilities.

Integration method:

```
pod "TCMPPExtLBS"
```

How to use:

Add following lines to the info.plist file:

`Privacy - Location Always and When In Use Usage Description`

`Privacy - Location Always Usage Description`

`Privacy - Location Usage Description`

`Privacy - Location When In Use Usage Description`

`Privacy - Motion Usage Description`

API list:

API name	API description
Location	Refer to Location.
Map	Refer to Map.
Compass, accelerometer, device motion, gyroscope	Refer to Refer to Compass, Accelerometer, Device Motion, Gyroscope.

TCMPPExtMDNS

TCMPPExtMDNS module provides the capabilities to communicate over a LAN.

Integration method:

```
pod "TCMPPExtMDNS"
```

How to use:

Add following lines to the info.plist file:

`Privacy - Local Network Usage Description`

`Privacy - Bonjour services`

API list:

API name	API description
mDNS	Refer to mDNS.

TCMPPExtNetwork

TCMPPExtNetwork module provides the capabilities for TCP/UDP communication.

Integration method:

```
pod "TCMPPExtNetwork"
```

How to use:

Add following lines to the info.plist file:

`App Transport Security Settings`

`Allow Arbitrary Loads - YES`

`Privacy - Bonjour services`

API list:

API name	API description
TCP	Refer to TCP Communication.
UDP	Refer to UDP Communication.

TCMPPExtMp3Encoder

TCMPPExtMp3Encoder module provides the capabilities to save files in MP3 format when using RecorderManager.

Integration method:

```
pod "TCMPPExtMp3Encoder"
```

How to use:

The MP3 saving feature provided by the SAS depends on the Lame library. The Lame library is open-source and licensed under the **GNU Library or Lesser General Public License version 2.0 (LGPLv2), GNU General Public License version 2.0 (GPLv2)** For more information, see: LAME. You can integrate it as needed.

TCMPPExtMiniGame

TCMPPExtMiniGame module provides the capabilities to run mini games.

Integration method:

```
pod "TCMPPExtMiniGame"
```

How to use:

Note that TCMPPExtMiniGame currently only supports running and debugging on real devices. It does not support running on simulators.

TCSASExtGoogleAds

TCSASExtGoogleAds supports Google AdMob ad loading.

Integration method:

```
pod "TCSASExtGoogleAds"
```

How to use:

Note that this feature depends on AdMob. Please integrate it according to the official documentation first. For more information, see Get Started**.**

TCSASBaseLib

TCSASBaseLib provides preloading of the base library to reduce the waiting time when the mini program is opened for the first time.

Integration method:

```
pod "TCSASBaseLib"
```

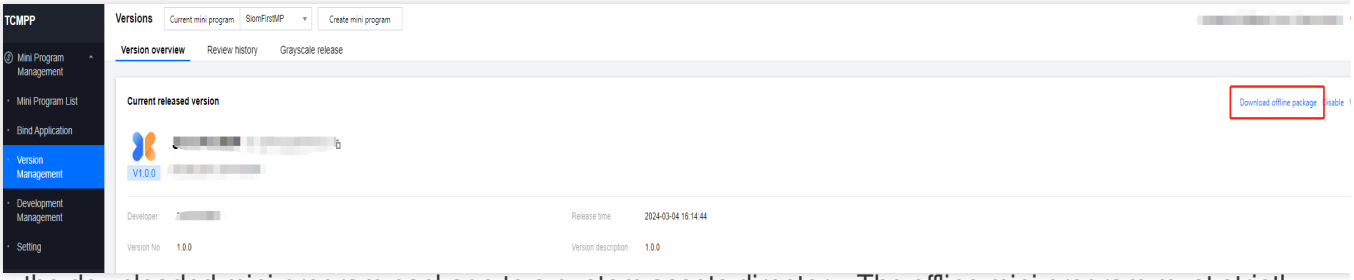
Preloading Offline Mini Programs

Last updated：2024-11-21 18:05:37

Offline mini programs are embedded within the host app. You need to download the mini program package from the console, import it into the host app project, and package its together with the app. During app usage, users can open an embedded mini program without needing to download it from the backend, allowing it to run even without an internet connection.

Preloading process

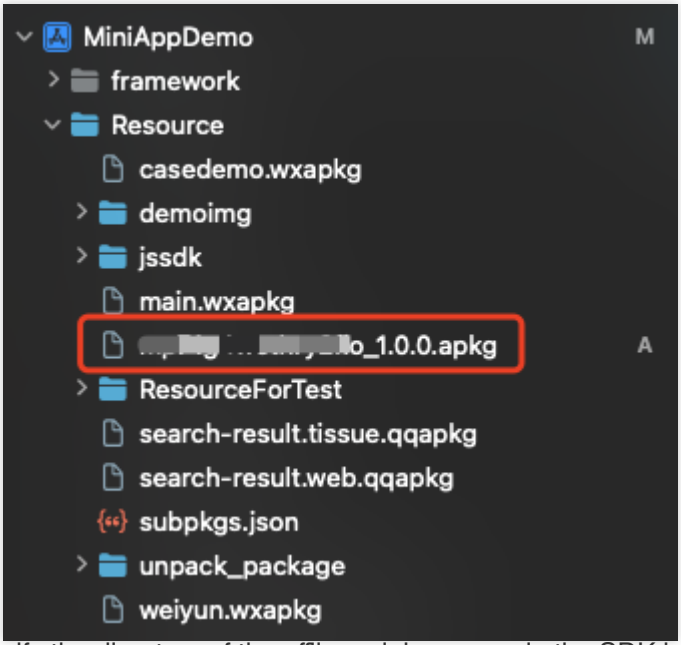
1. Download the required mini program from the console.



2. Copy the downloaded mini program package to a custom assets directory. The offline mini program must strictly follow the naming convention and cannot be arbitrarily modified.

Note:

Naming convention for mini programs: {miniAppId}_{miniAppVersion}.apkg



3. Specify the directory of the offline mini program in the SDK initialization configuration.

```
[[TMFMiniAppSDKManager sharedInstance] setOffLineApkgsPath:[[[NSBundle mainBundle]
```

Notes

Offline mini programs must go through the mini program binding and releasing process. Only mini programs in the online state can download offline packages;
Offline mini programs adhere to version management logic such as new version releases, version rollbacks, and removing. If the online version differs from the preloaded version, the client will fetch the online version;
If a mini program is removed, the corresponding offline mini program preloaded in the app will also become unusable.

iOS FAQs

Last updated：2025-03-17 15:45:41

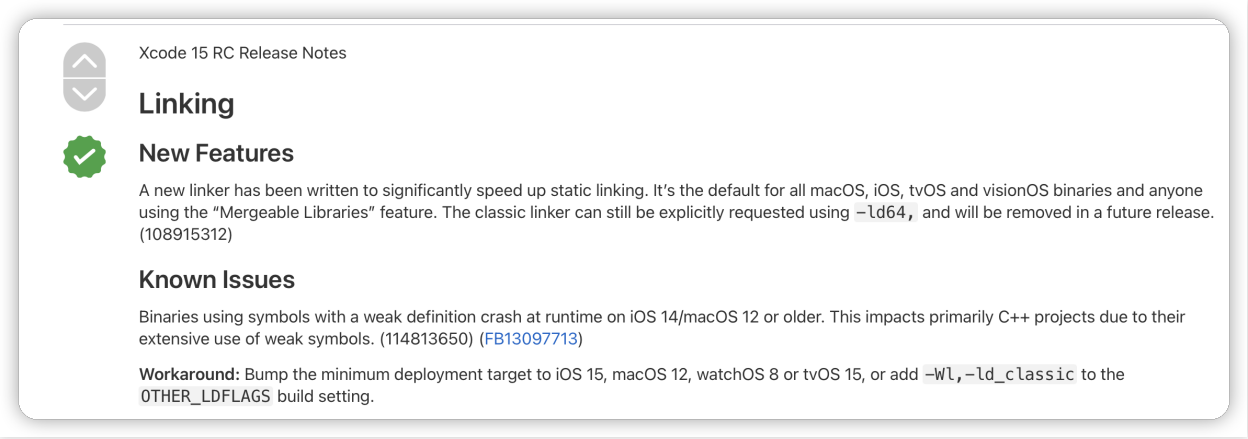
Integration FAQs

What is the minimum configuration required to integrate the SDK?

The mini program SDK supports iOS 9 later versions. The versions earlier than iOS 9 are not supported.

Crashes on iOS 12 and earlier versions after compiling with XCode 15

Add the -Wl,-ld_classic configuration in the project compilation options.



What is the impact on the installation package size after integrating the SDK?

The SDK uses a core library and extension libraries, allowing users to integrate the SDK as needed. Integrating only the core library impacts the overall installation package size by approximately 4 MB.

FAQs during usage

Troubleshooting mini program startup failures

If the mini program fails to start, it could be due to the following reasons:
Reason 1: Incorrect configuration file path. If the configuration file is in a sub-directory, you need to append the directory path, e.g., server/tcsas-ios-configurations.json;
Reason 2: Modifying the mini program configuration file content is not allowed; otherwise, the mini program will not run properly;

Reason 3: The bundleId in the configuration file must match the application's bundleId. Otherwise, the app will fail to run because the bundleId in the configuration file is verified during initialization.

Is it supported to completely hide the navigation bar on a page of a mini program? If it is supported, how to implement it?

A: Support.

The navigationStyle of mini programs like WeChat, Alipay, Baidu, etc. has two values of default/custom, while the mini program SDK adds a new hide on top of that, that is, it has three values of default/custom/hide.

So on the page where you need to hide the navigation bar, just set navigationStyle to hide.

How do I set up grey scale release conditions?

The host application developer can set the user's attribute information (region or account number) through the method provided by the SDK, which is convenient for use in scenarios such as grey-scale push.

Set User ID

```
// Report user ID, if parameter is null it is considered unbound
// Report user ID, if parameter is null it is considered unbound.
// @param customisedUserID User customised user ID - customised user ID
- (void)updateCustomizedUserID:(nullable NSString *)customizedUserID;
```

Setting location information

```
// Report location information
// Report location information
// @param country Country - country
// @param province - province
// @param city - city
- (void)updateAreaInfoWithCountry:(nullable NSString *)country Province:(nullable N
```

```
[[TMFMiniAppSDKManager sharedInstance] updateCustomizedUserID:@"abc123"];
[[TMFMiniAppSDKManager shared instance] updateAreaInfoWithCountry:@"China" Province
```

Does it support uploading to a tri-party service when data is reported?

Support APP to report mini program data to any service, see Logging and event reporting.

Does mini program support data isolation for different login users?

Yes.

The SDK supports segregated storage of mini program data according to different logged-in users for data protection and business logic confusion, which requires the developer to implement the return of AppUID in the agent, and the SDK will store the data according to the AppUID.

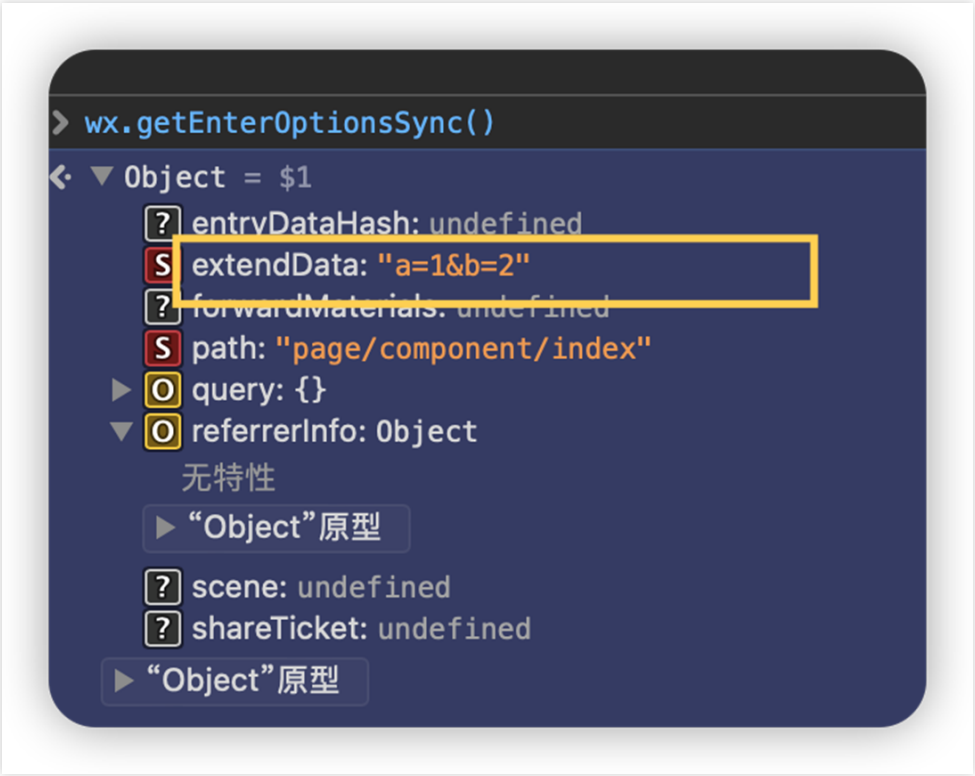
```
/**
 * @brief Get the current user account identifier of the SDK host platform, usually

 * Note: Returning nil will invalidate some caches within the SDK. If not logged in
 */

- (NSString *_Nonnull)getAppUID;
```

How do I pass parameters when Mini programs open?

Mini programs can be opened by specifying the paramsStr parameter:

```
[[TMFMiniAppSDKManager sharedInstance] startUpMiniAppWithAppID:@"mpbz0uqwzddj5zbt"
```

In Mini programs, you can get the parameters passed in when opening in the extendData field of wx.getEnterOptionsSync:



Debugging Related

How to debug a mini program?

After integrating the mini program SDK and running the mini program in Xcode (apps packaged using the develop profile can also be viewed using Safari on your computer), you can open Safari on your computer, and then in the toolbar Develop, select the running emulator or real device, you can select the list of pages opened by the mini program, and by selecting the currently opened page, you can review page elements, view network calls, and some logs.

After ios16.4, because of system limitations, you need to turn on inspectable to support safari development mode, you can implement the relevant interfaces in the proxy class and return YES.

```
// Whether to enable checkable after ios16.4, you can develop and debug via safari
// Whether or not inspectable is enabled after ios16.4, you can develop and debug v
- (BOOL)inspectableEnabled;
```

How to open vConsole?

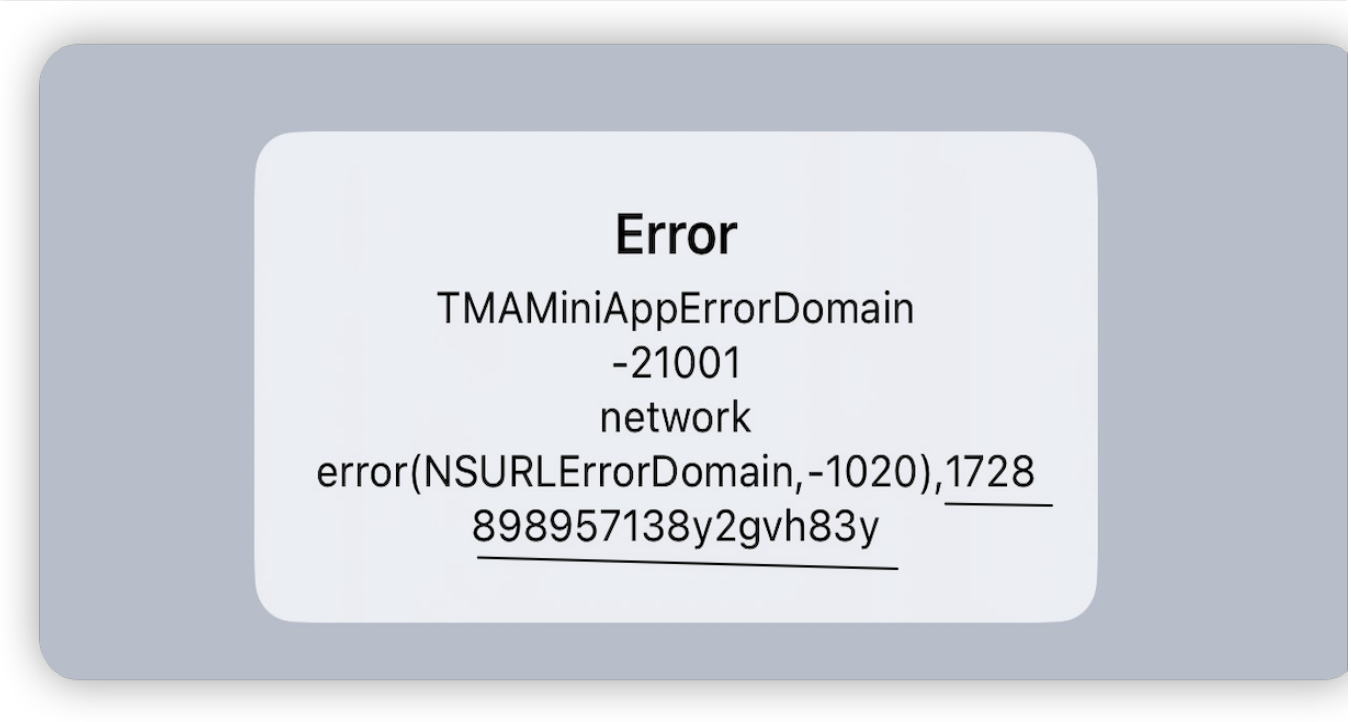
The development version and preview version of mini programs support to open the debug switch in the capsule menu to view the VConsole information, the official version of the mini program can not open the VConsole, you need to open the VConsole through the mini program internal call wx.setEnableDebug.

The SDK also supports to open the vConsole mode of all mini programs automatically in the app settings, you need to implement the related interface in the proxy class.

```
// Whether debugging is enabled
// Whether debugging is enabled
- (BOOL)vConsoleEnabled;
```

How to troubleshoot when a mini program opens abnormally?

When opening a mini program, you can get the error code and error description in the mini program callback interface. The error description contains the traceId of the current request. It is strongly recommended that you save the error information to facilitate troubleshooting.



iOS Error Codes

Last updated：2024-11-21 18:06:21

Error codes for opening mini program

Returns error code description

```
// Initialization information is incorrect or not initialized
TMAMiniAppErrorConfigInitError = -21000,

// The networkerror
TMAMiniAppErrorNetworkError = -21001,

// The server returns an error code
TMAMiniAppErrorResponseCodeError= -21002,

// The server returns an empty response
TMAMiniAppErrorResponseDataNull = -21003,

// The mini program version does not exist
TMAMiniAppErrorVersionNotExist= -21005,

// The mini program has been removed from the shelves
TMAMiniAppErrorForbidden= -21006,

//The frequency of specific interface calls is too high
TMAMiniAppCmdIDRequestFrequent = -21011,

//Interface calls are too frequent
TMAMiniAppRequestFrequent = -21012,

// Unsupported link
TMAMiniAppErrorErrorLink = -21101,

// TCMPPExtScanCode extension library not integrated
// Unsupported link
TMAMiniAppErrorNoScanCodeSDK= -21102,

// delegate is not implemented
TMAMiniAppErrorNotImplemented = -21103,

// Mini program base library download failed
TMAMiniAppErrorJSSDKDownloadFail= -22001,
```

```
// Mini program base library decompression failed
TMAMiniAppErrorJSSDKUnzipFail = -22002,

// Mini program base library does not need to be upgraded
TMAMiniAppErrorJSSDKNeedUpgrade = -22003,

// Mini program basic library verification failed
TMAMiniAppErrorJSSDKVerifyFail= -22004,

TMAMiniAppErrorResponseInvalid= -22010,

// UnPack mini program Apkg package failed
TMAMiniAppErrorUnpackApkgFail = -22011,

TMAMiniAppErrorConsistencyInvalid = -22012,

TMAMiniAppErrorHTTPStatusInvalid= -22013,

// The download storage path of the mini program Apkg package is wrong
TMAMiniAppErrorCreatePathFail = -22014,

// The download storage path of the mini program Apkg package is wrong
TMAMiniAppErrorValidateFileFail = -22015,

// The device system does not support mini programs
TMAMiniAppErrorSystemNotSupport = -22017,

// AppInfo error when starting the mini program
TMAMiniAppErrorAppInfoError = -22018,

// The mini program is already being displayed
TMAMiniAppErrorMiniAppIsShowing = -22019,

// The mini program view is an invalid view
TMAMiniAppErrorParentVCInvalid= -22020,

// Application error when starting the mini program
TMAMiniAppErrorApplicationError = -22021,

// The agent for starting the mini program is not implemented

TMAMiniAppErrorDelegateError= -22022,
TMAMiniAppErrorNilApplication = -22023,
TMAMiniAppErrorUserNotLoggedIn= -22024,
TMAMiniAppErrorFlutterResourceInvalid = -22025,
```

```
// An exception occurs when the mini program reads the resource file (for example,
TMAMiniAppErrorReadPackageFail= -22027,

// Download file md5 verification failed
TMAMiniAppErrorFileMd5VerifyFail = -22029,

// Failed to copy domain name configuration file to target location
TMAMiniAppErrorFCopyDomainsFileFail = -22030,

// MAWebView white screen error
TMAMiniAppErrorWebViewNoCompositingView = -23001,
TMAMiniAppErrorWebViewJSError = -23002,
TMAMiniAppErrorWebViewDidTerminate= -23003,
TMAMiniAppErrorWebViewScriptAgainError= -23004,
TMAMiniAppErrorWebViewPageNotFound= -23005,
TMAMiniAppErrorWebViewApplicationNotFound = -23006,

TMAMiniAppErrorHotlaunchParamInvalid= -23007,
TMAMiniAppErrorWebViewRenderBlank = -23008,

// No available cloud resources to use
TMAMiniAppErrorResourceLimited = -25001,
TMAMiniAppErrorResourceOverflow = -25002
```

iOS Privacy Compliance

Last updated：2025-03-20 18:08:15

System permissions information involved in the mini program SDK:

SDK Name	System permissions involved	Permission usage description
TCMPPSDK	NSPhotoLibraryAddUsageDescription NSCameraUsageDescription NSMicrophoneUsageDescription	Used to realize the functions of taking pictures, recording audio, saving
TCMPPExtMedia	NSPhotoLibraryUsageDescription	Used to select photos and videos
TCMPPExtScanCode	NSCameraUsageDescription	Used to scan QR codes
TCMPPExtQMap	NSLocationAlwaysAndWhenInUseUsageDescription NSLocationUsageDescription NSLocationAlwaysUsageDescription NSLocationWhenInUseUsageDescription	Used to locate and use Tencent map services
TCMPPExtBaiduMap	NSLocationAlwaysAndWhenInUseUsageDescription NSLocationUsageDescription NSLocationAlwaysUsageDescription NSLocationWhenInUseUsageDescription	Used for positioning and using Baidu map service
TCMPPExtAMap	NSLocationAlwaysAndWhenInUseUsageDescription NSLocationUsageDescription NSLocationAlwaysUsageDescription NSLocationWhenInUseUsageDescription	Used for positioning and using Amap service
TCMPPExtLive	NSCameraUsageDescription NSMicrophoneUsageDescription	Used for implementing real-time audio and video recording function
TCMPPExtAuthentication	NSFaceIDUsageDescription	Used for using device biometric authentication function
TCMPPExtBLE	NSBluetoothAlwaysUsageDescription NSBluetoothPeripheralUsageDescription	Used to implement Bluetooth related

		functions, such as Bluetooth device connection, etc.
TCMPPExtCalendar	NSCalendarsUsageDescription NSRemindersUsageDescription	Used to implement date selection and adding reminders
TCMPPExtClipBoard	-	Used to access the pasteboard
TCMPPExtContact	NSContactsUsageDescription	Used to obtain and add contacts
TCMPPExtLBS	NSLocationAlwaysAndWhenInUseUsageDescription NSLocationUsageDescription NSLocationAlwaysUsageDescription NSLocationWhenInUseUsageDescription NSMotionUsageDescription	Used to implement positioning, system maps and motion-related functions
TCMPPExtMDNS	NSBonjourServices	Used to provide LAN communication capabilities
TCMPPExtNetwork	NSAppTransportSecurity NSAllowsArbitraryLoads NSBonjourServices	Used to provide TCP/UDP communication capabilities
TCMPPExtMp3Encoder	-	Used for audio MP3 format encoding

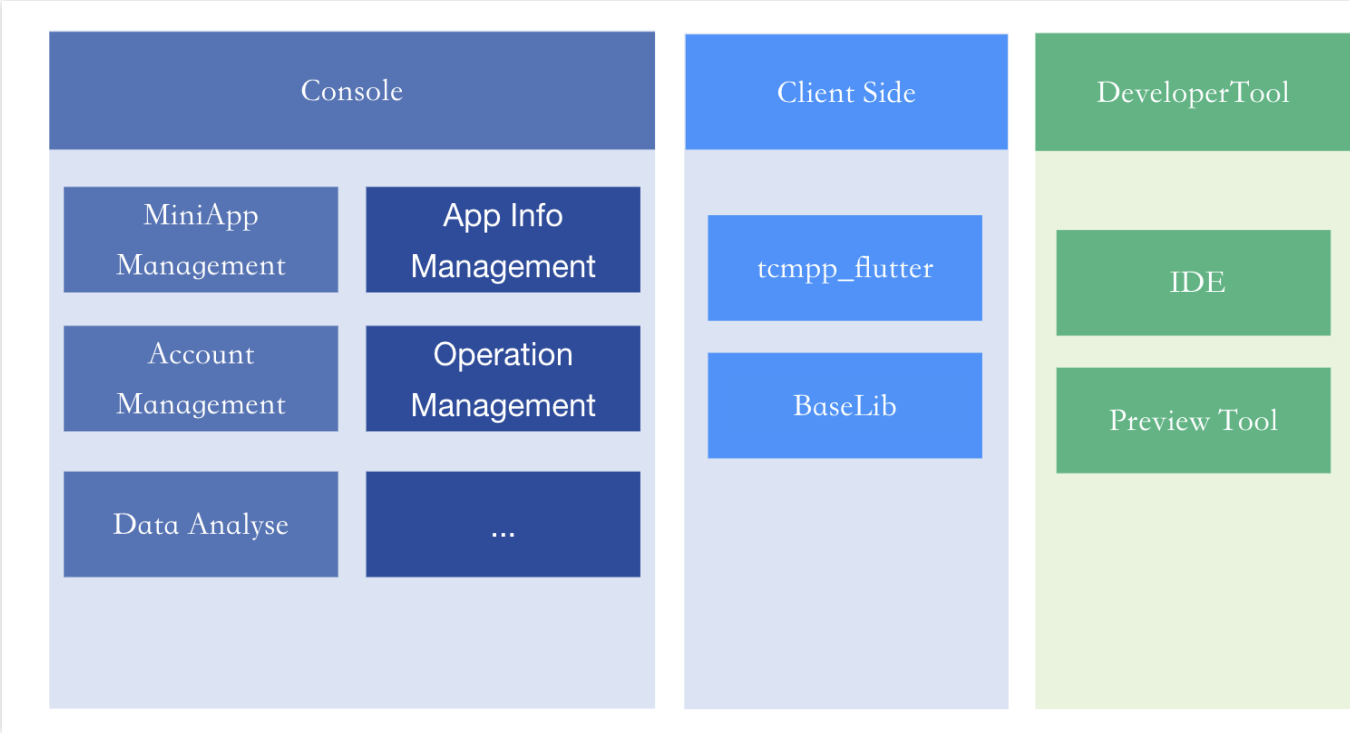
For more information, see Privacy Compliance.

Flutter
Flutter SDK Description
SDK Overview

Last updated：2025-02-10 16:16:06

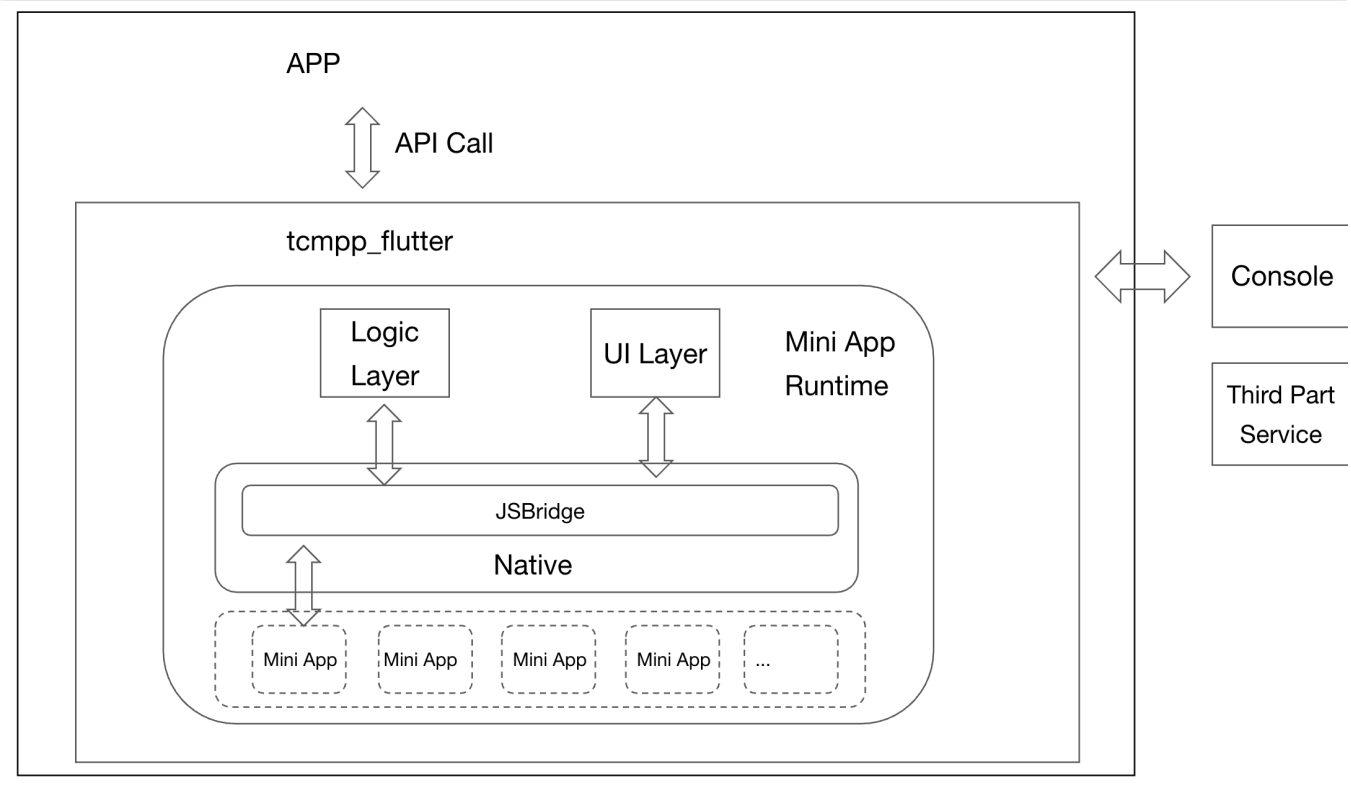
Product Introduction

The product consists of three parts: the console, tcmpp_flutter, and DevTools.



Console: The product console provides capabilities such as mini program management, client app management, and mini program operations management.
Client: The mobile client app that integrates tcmpp_flutter. tcmpp_flutter provides a runtime environment for mini programs in client apps.
DevTools: Use IDE for development, debugging, and version submission, and the mini program preview assistant to preview on mobile.

How tcmpp_flutter works



tcmpp_flutter provides a runtime environment for mini programs in client apps. It communicates with the product backend to retrieve, load, and run mini-programs within its runtime environment.

To reduce package size and minimize system permissions, tcmpp_flutter is split into a core library (tcmpp_flutter) and extension libraries (tcmpp_flutter_xxx). Most features are available with just the core library, which adds about 4 MB to the final app package and requires no extra system permissions.

For details about the extension library, see Flutter - Extensions.

SDK Quick Integration

Last updated：2025-05-26 15:26:15

Example for integration

You can download stable versions of tcmpp_flutter.

Prerequisites

Environment requirements

sdk: '>=3.0.0 <4.0.0'

flutter: '>=3.3.0'

Plugin dependencies

plugin_platform_interface

flutter_plugin_android_lifecycle

Integration method

Add the core plugin from Pub.dev

1. To add dependencies, you first need to add the plugin frompub.dev Open the pubspec.yaml file in the app folder and add tcmpp_flutter: ${version}.

2. Install dependencies
From the terminal: Run flutter pub get.
From VS Code: Click **Get Packages** located in right side of the action ribbon at the top of pubspec.yaml indicated by the Download icon.
From Android Studio/IntelliJ: Click **Pub get** in the action ribbon at the top of pubspec.yaml.
For details, seePackages & plugins > Using packages.

Obtain the configuration file

To initialize the mini program SDK, first obtain the SDK configuration file from the mini program console.
Log in to the console, and click **Create application**.

In the pop-up window, fill in the app information. In the integration platform section, check both the iOS and Android platforms. Enter the bundle ID or application ID in the package name field.

Click **Next** to download the Android/iOS configuration files.

Add the configuration file

1. Place the configuration file in the Flutter project.
2. Add a file named tcsas-plugin-settings.json in the root directory of the project.
3. Edit the pubspec.yaml file to include the above files as a Flutter resource.

In the **tcsas-plugin-settings.json** file, you can configure the SDK settings as needed.

Android:

Field	Type	Description
configAssetsName	string	Path to the configuration file in Android
debug	bool	Enable SDK debugging and logging
enableX5Core	bool	Whether to use Tencent Browsing Service (X5 kernel) as the mini program runtime
x5LicenseKey	string	Tencent Browsing Service license key if the runtime is enabled
X5DocLicenseKey	string	Tencent Browsing Service documentation license key if the runtime is enabled
x5Core32Url	string	Download URL for the 32-bit Tencent Browsing Service runtime
x5Core64Url	string	Download URL for the 64-bit Tencent Browsing Service runtime

iOS:

Field	Type	Description
configAssetsName	string	Path to the configuration file in iOS
debug	bool	Enable SDK debugging
logEnable	bool	Enable SDK logging
inspectableEnabled	bool	Enable inspectable feature for iOS 16.4 and later, allowing mini program debugging via Safari

General:

Field	Type	Description
appName	string	Host app name, mainly for copyright notices in the mini programs
appVersion	string	Host app version, mainly for copyright notices in the mini programs
assetPathOfPresets	string	Directory for preloaded offline mini programs, requires a relative path under Flutter assets

Example:

```
{
  "android": {
    "configAssetsName": "tcsas-android-configurations.json",
    "debug": true,
    "enableX5Core": true,
    "x5LicenseKey": "",
    "x5DocLicenseKey": "",
    "x5Core32Url": "",
    "x5Core64Url": ""
  },
  "ios": {
    "configAssetsName": "tcsas-ios-configurations.json",
    "debug": true,
    "logEnable": true,
    "inspectableEnabled": true
  },
  "common":{
    "appName": "tcmpp-flutter",
    "appVersion": "1.0.0",
    "assetPathOfPresets": "res/offline"
  }
}
```

In Android, go to android > app > build.gradle in your Flutter project, and the applicationId can be found in android > defaultConfig.

In iOS, go to ios > Runner.xcodeproj > project.pbxproj in your Flutter project, and search for PRODUCT_BUNDLE_IDENTIFIER.

Platform-specific

For Android

1. Navigate to the android > app directory and open the build.gradle file for editing.
2. In the android > defaultConfig section, set the minSdkVersion to at least 21.

```
minSdkVersion 21
```

3. In the android > defaultConfig section, add packagingOptions.

```
packagingOptions {
   pickFirst 'lib/arm64-v8a/libc++_shared.so'
   pickFirst 'lib/armeabi/libc++_shared.so'
   pickFirst 'lib/armeabi-v7a/libc++_shared.so'
   pickFirst 'lib/arm64-v8a/libmarsxlog.so'
   pickFirst 'lib/armeabi/libmarsxlog.so'
   pickFirst 'lib/armeabi-v7a/libmarsxlog.so'
   pickFirst 'lib/arm64-v8a/libv8jni.so'
}
```

For iOS

Add the source code

Add the source code in the Podfile located in the ios directory.

```
source 'https://e.coding.net/tcmpp-work/tcmpp/tcmpp-repo.git' source
'https://github.com/CocoaPods/Specs.git'
```

Install pods

Change directory to the **ios** and execute pod install.

```
pod install
```

Open a mini program

Import 'package:tcmpp_flutter/tcmpp_flutter.dart' using the API.

Create a TcmppFlutter object and use it to call the API to open a mini program:

Common SDK Integration Issues

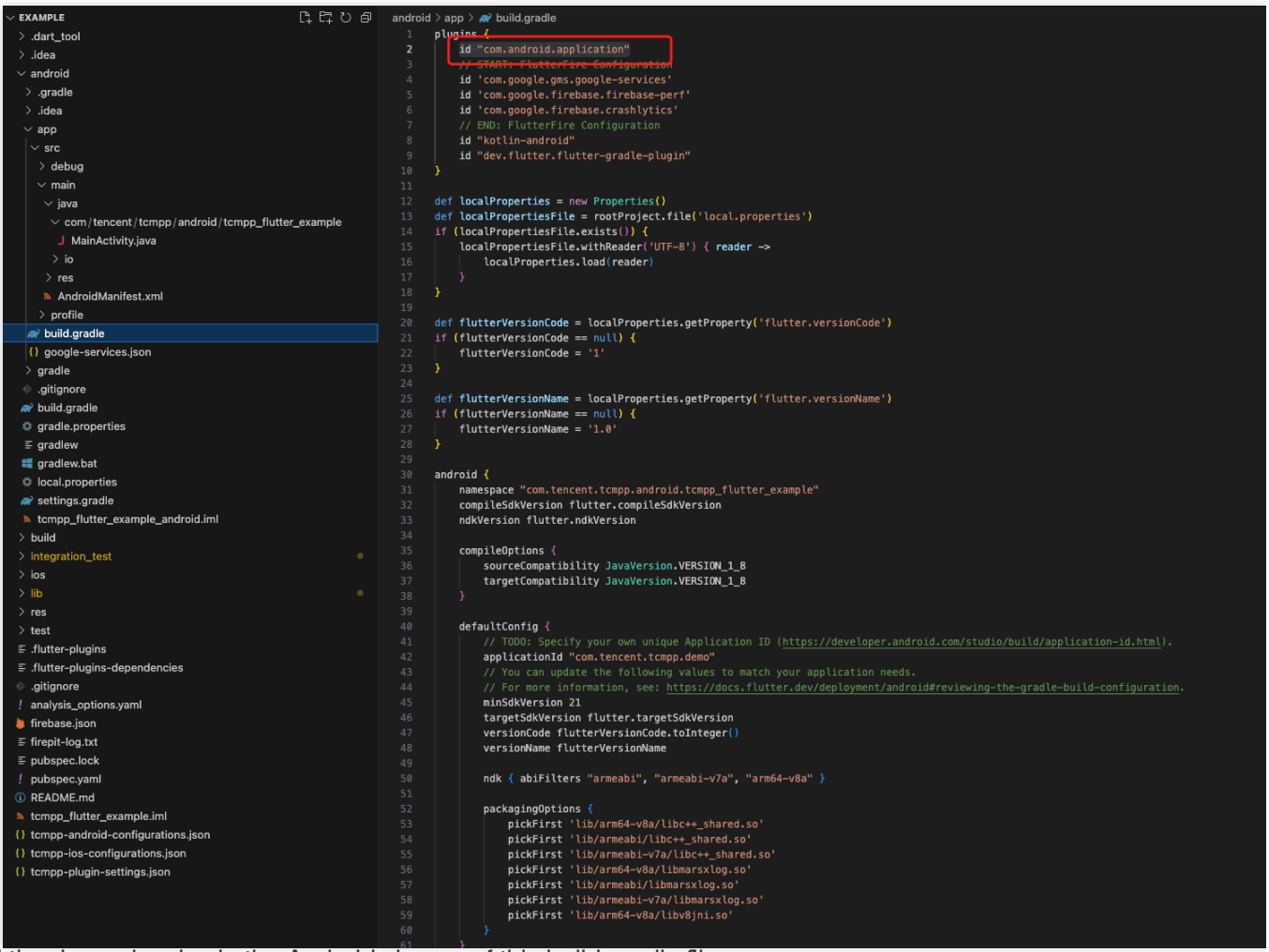
Last updated：2025-02-10 16:22:58

Crash issues with Firebase Flutter plugin

In Android, the Firebase Flutter plugin does not support multi-process handling by default, which may cause the mini program subprocess to crash. To use Flutter plugins with Firebase Flutter plugin, you need to initialize Firebase in the mini program subprocess.

Adding the dependency

1. Go to the root directory of the Flutter project. In the Flutter project, search `id "com.android.application"`
The build.gradle file containing this string is located in the root directory of your Flutter Android project.



2. Add the dependencies in the Android closure of this build.gradle file:

```
dependencies {
            compileOnly 'com.tencent.tcmpp.android:mini_core:+'
```

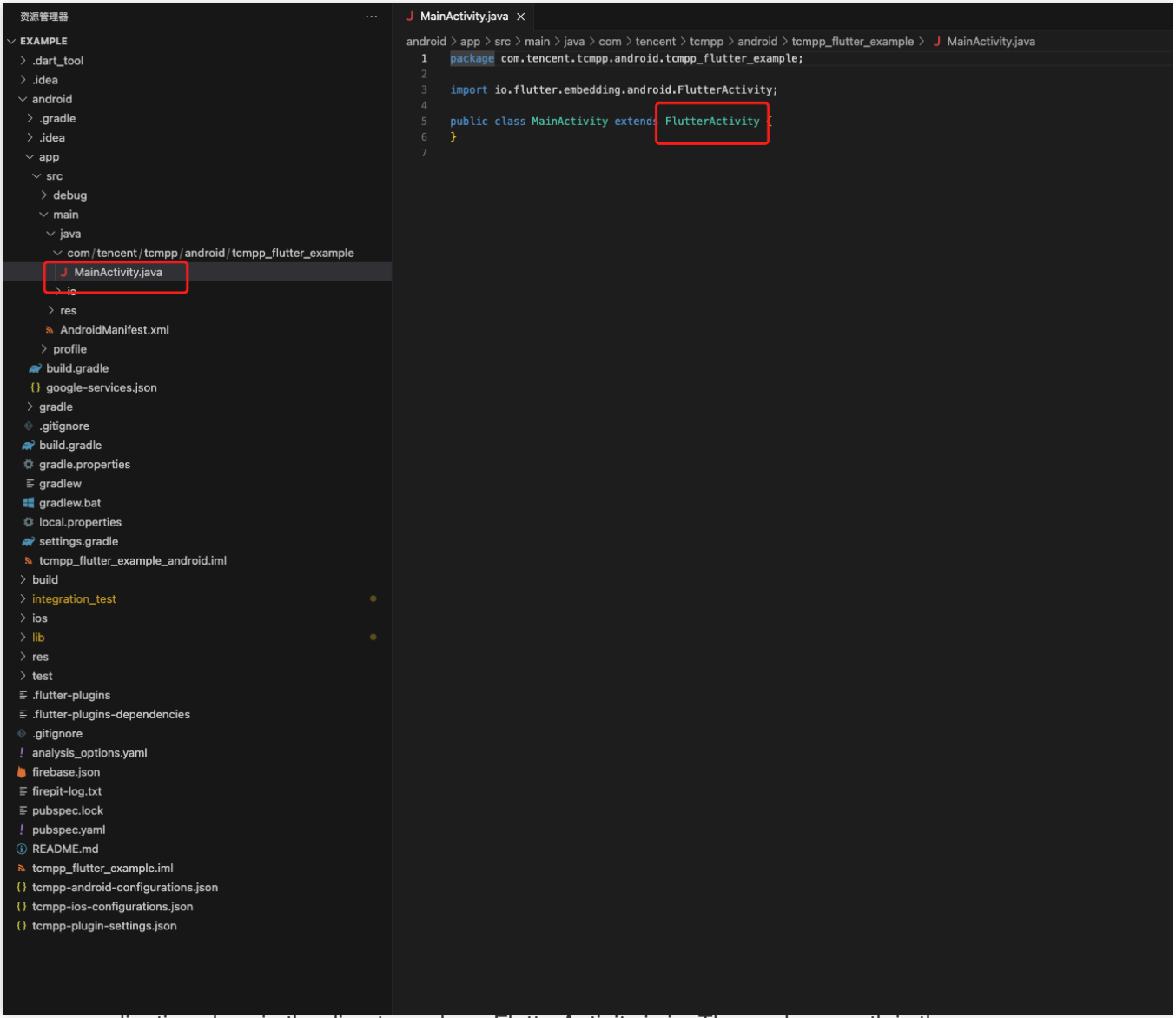
```
            compileOnly 'com.google.firebase:firebase-common:+'
        }
```

Creating an application class

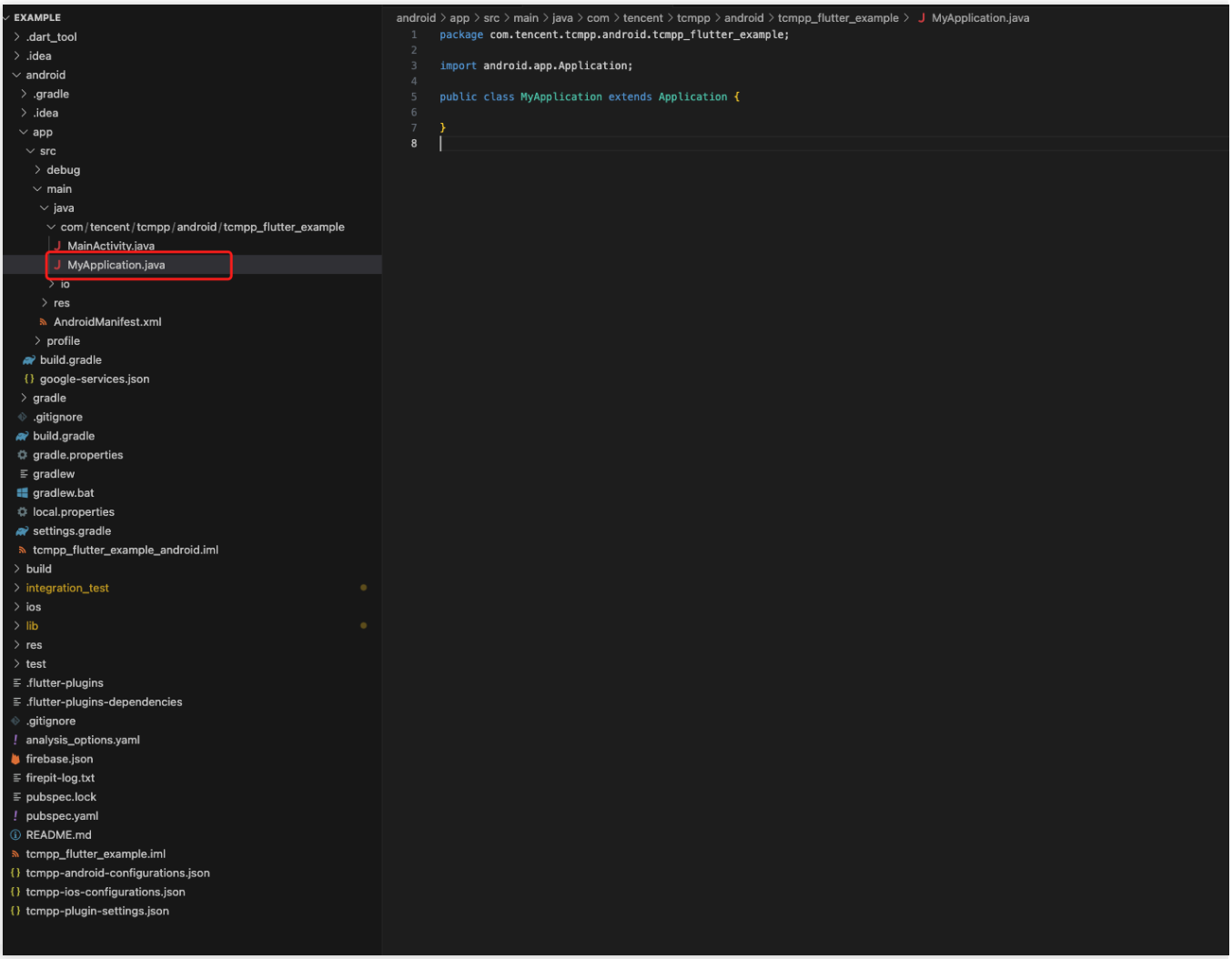
Note：

Skip this step if the Flutter project already has a custom Android application class.

In the src directory, locate FlutterActivity:

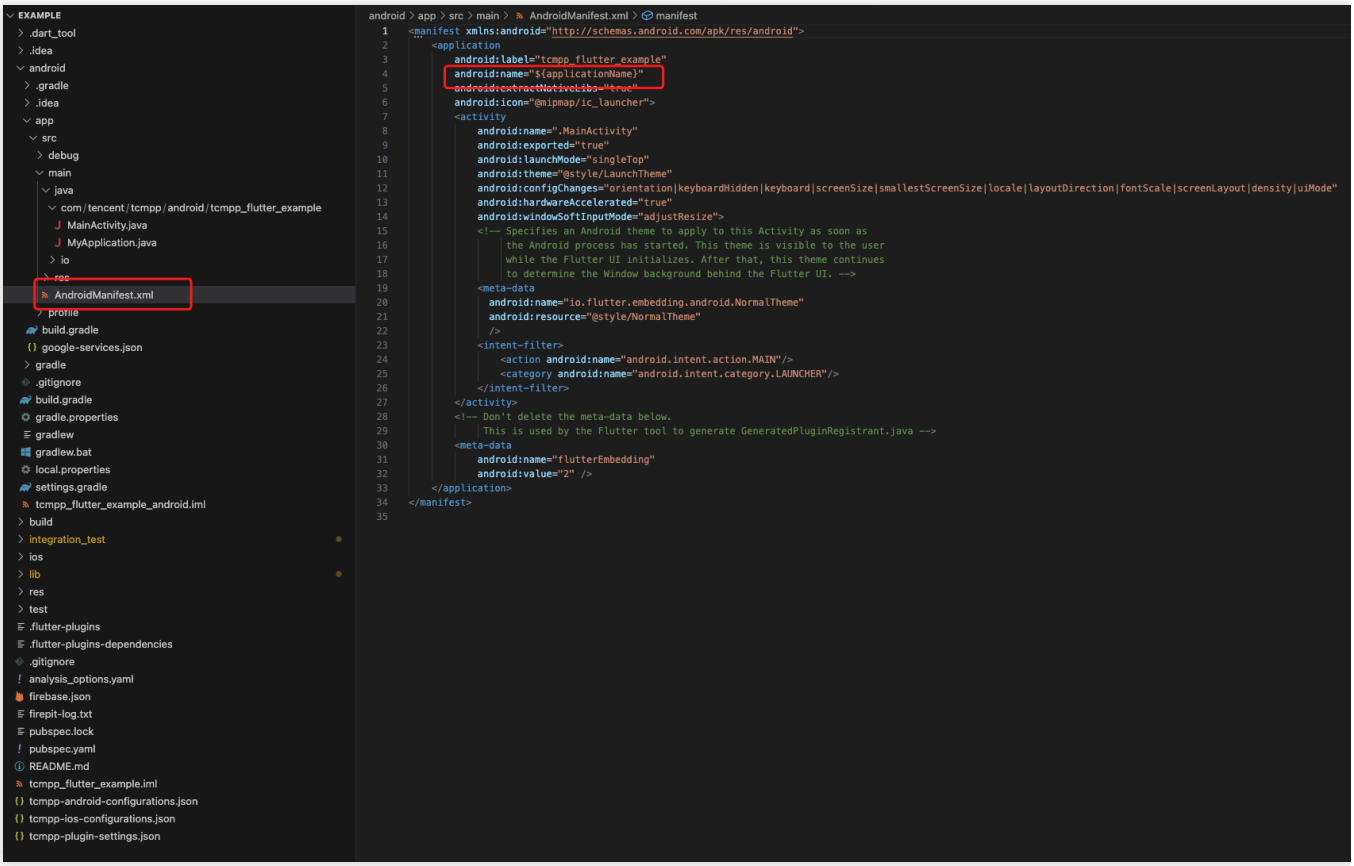


Create a new application class in the directory where FlutterActivity is in. The package path is the same as FlutterActivity:




```
import android.app.Application;
public class MyApplication extends Application {}
```

Open and edit the AndroidManifest.xml under src > main. In the <application> tag, change the android:name to match your application class:




```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"><application
```

Initializing Firebase in the subprocess

1. Override the onCreate function in your Android application class. Skip this step if it is already done.

```
public void onCreate() {
        super.onCreate();
    }
```

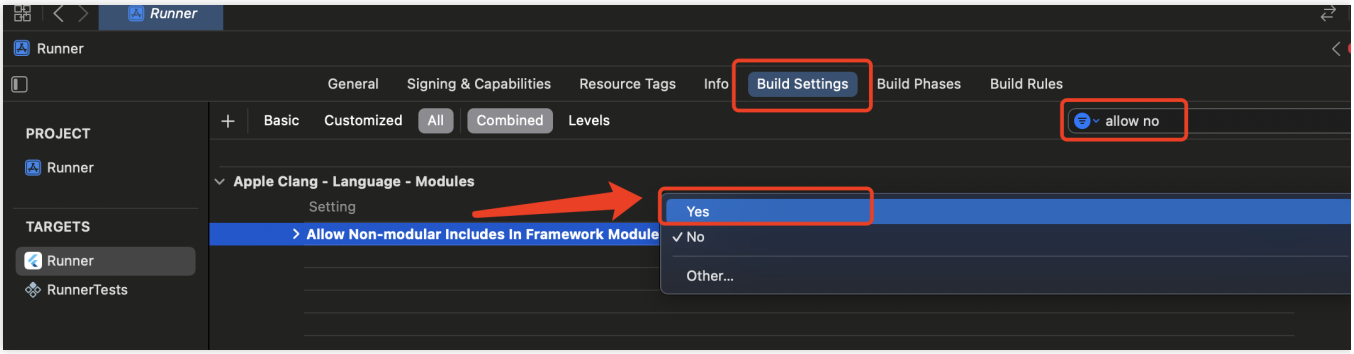
2. Add the Firebase initialization code for the subprocess.

```
import android.app.Application;
    import com.google.firebase.FirebaseApp;
    import com.tencent.tmf.mini.api.TmfMiniSDK;

    public class MyApplication extends Application {

        @Override
        public void onCreate() {
            super.onCreate();

            ...
```

```
            if (TmfMiniSDK.isMiniProcess(this)) {
                FirebaseApp.initializeApp(this);
            }

            ...
        }
    }
```

The header file cannot be found in compilation in iOS

Open the project with Xcode, select the right target and go to Build Settings. Enter "Allow Non-modular Includes In Framework Modules" in the search box and set it to "Yes":



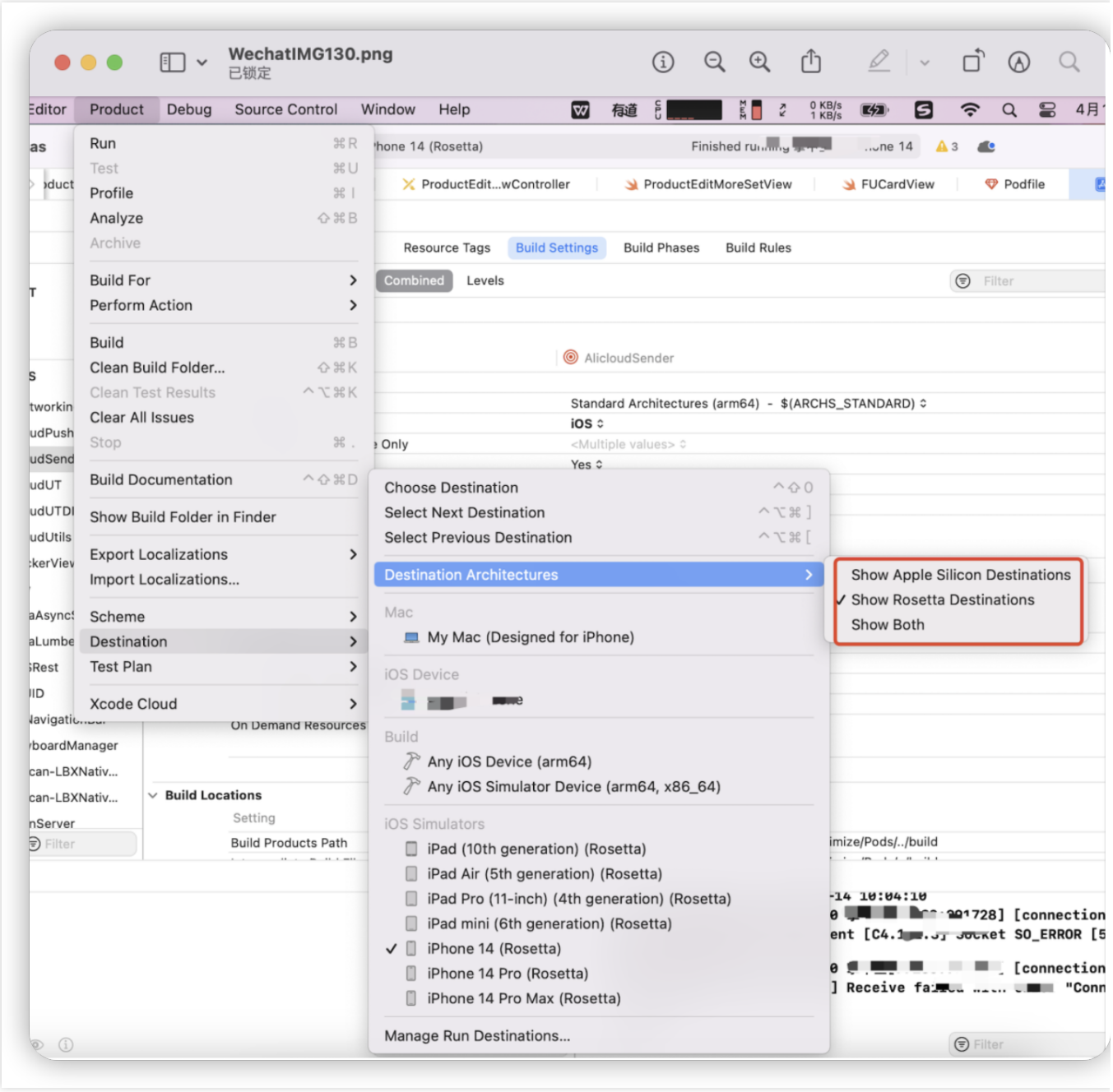
Apple M-series chip-based computer emulator issues

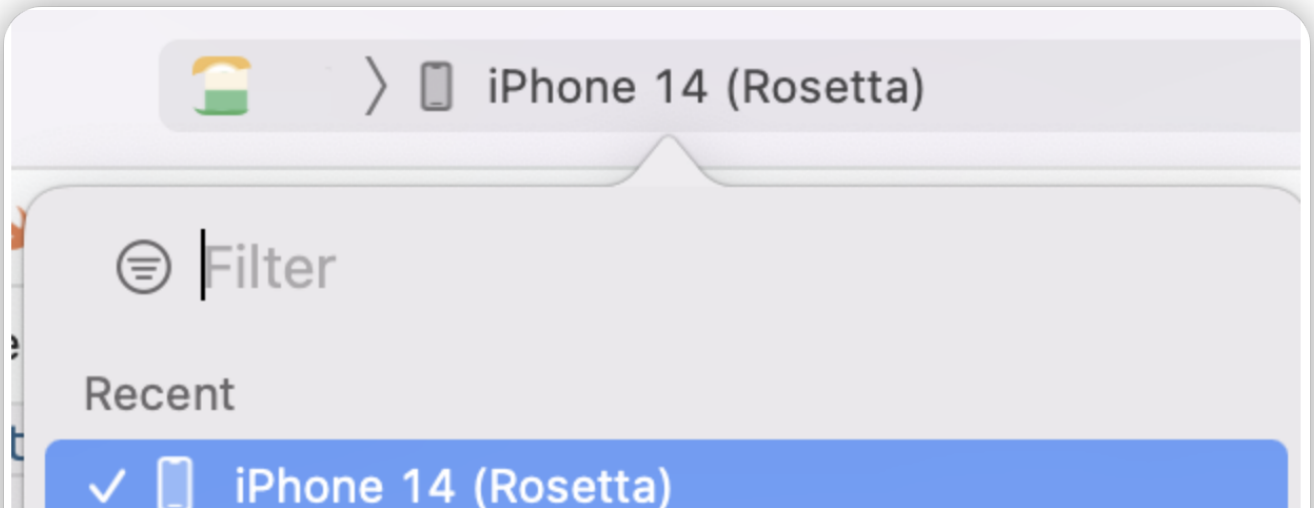
The current version of SDK need to run via Rosetta on the M-series chip-based computer emulator.

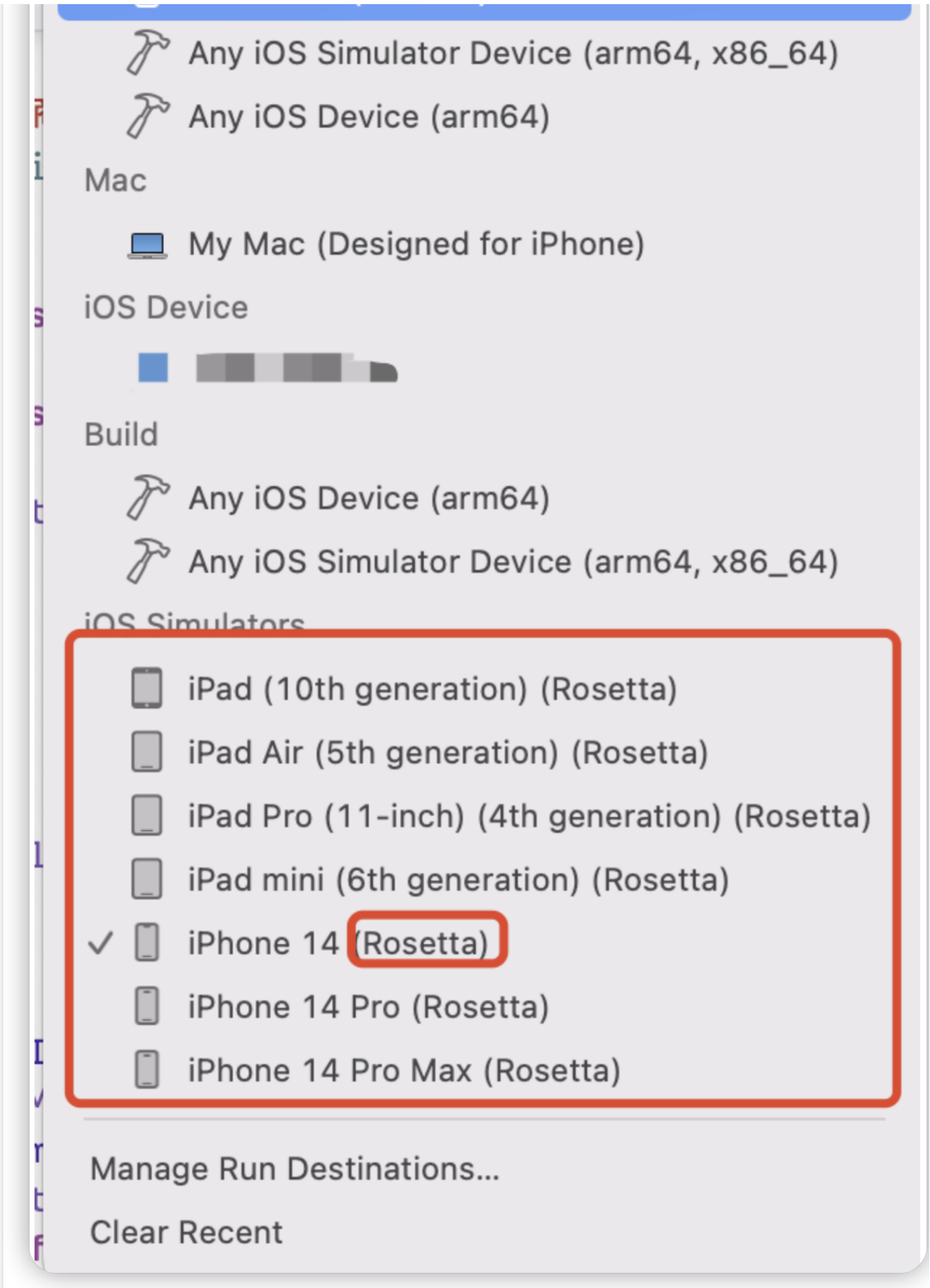
For versions earlier than Xcode14: Right-click Xcode -> Get Info, and check Open using Rosetta:

For versions later than Xcode14:

Open the project with Xcode -> Product -> Destination -> Destination Architecturesk. Choose Show Rosetta Destinations:


WechatIMG130.png
已锁定
Editor
Product
Debug
Source Control
Window
Help
Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For
Perform Action
Build ⌘B
Clean Build Folder... ⇧⌘K
Clean Test Results ^⌥⌘K
Clear All Issues
Stop ⌘.
Build Documentation ^⇧⌘D
Show Build Folder in Finder
Export Localizations
Import Localizations...
Scheme
Destination
Test Plan
Xcode Cloud
Resource Tags
Build Settings
Build Phases
Build Rules
Combined
Levels
Filter
AlicloudSender
Standard Architectures (arm64) - $(ARCHS_STANDARD)
iOS
Choose Destination ^⇧0
Select Next Destination ^⌥⌘]
Select Previous Destination ^⌥⌘[
Destination Architectures
Show Apple Silicon Destinations
Show Rosetta Destinations
Show Both
Mac
My Mac (Designed for iPhone)
iOS Device
Build
Any iOS Device (arm64)
Any iOS Simulator Device (arm64, x86_64)
iOS Simulators
iPad (10th generation) (Rosetta)
iPad Air (5th generation) (Rosetta)
iPad Pro (11-inch) (4th generation) (Rosetta)
iPad mini (6th generation) (Rosetta)
iPhone 14 (Rosetta)
iPhone 14 Pro (Rosetta)
iPhone 14 Pro Max (Rosetta)
Manage Run Destinations...
Build Locations
Setting
Build Products Path


iPhone 14 (Rosetta)
Filter
Recent
iPhone 14 (Rosetta)


Any iOS Simulator Device (arm64, x86_64)
Any iOS Device (arm64)
Mac
My Mac (Designed for iPhone)
iOS Device
Build
Any iOS Device (arm64)
Any iOS Simulator Device (arm64, x86_64)
iOS Simulators
iPad (10th generation) (Rosetta)
iPad Air (5th generation) (Rosetta)
iPad Pro (11-inch) (4th generation) (Rosetta)
iPad mini (6th generation) (Rosetta)
iPhone 14 (Rosetta)
iPhone 14 Pro (Rosetta)
iPhone 14 Pro Max (Rosetta)
Manage Run Destinations...
Clear Recent

Flutter API
Mini Program Management API
Opening the Mini Program

Last updated：2024-08-06 16:00:50

Opening the mini program

When opening the mini program, the system will check whether there is a locally cached one. If not, it will automatically download the mini program from the remote server and then open it. If a cached version is available, the local mini program will be opened first while the system checks in the background for any new versions on the server.

Note：

If a new version is available, it will be downloaded, and the next time you open the mini program, the updated version will be used.

Start up the mini program with the given appId

appId: The ID of the mini program

Options: Startup options for the mini program

```
Future<void> startMiniAppWithId(String appId, MiniStartOptions? options)
```

Sample code:

```
 _tcmppFlutterPlugin.startMiniAppWithId(appId!, null);
```

Startup options

```
class MiniStartOptions {
  /// The path to enter the mini program
  String? entryPath;

  /// Always update the mini program at startup
  bool? isForceUpdate;

  /// String parameters passed to the mini program at startup
  String? params;
}
```

Start up the mini program via the given link

Link: The URI to start up the mini program

options: The startup options for the mini program

```
Future<void> startMiniAppWithLink(String link, MiniStartOptions? options)
```

Sample code:

```
_tcmppFlutterPlugin.startMiniAppWithLink(appId!, null);
```

Closing the Mini Program

Last updated：2024-08-06 15:59:43

Note：

Closing the mini program will clear it in memory. When the mini program is reopened, it needs to be loaded into memory again, which can take some time.

Stop the mini program with the given ID.

appId: The ID of the mini program:

```
Future<void> stopMiniApp(String appId)
```

Sample code:

```
_tcmppFlutterPlugin.stopMiniApp(appId!)；
```

Stop all running mini programs:

```
Future<void> stopAllMiniApp()
```

Sample code:

```
_tcmppFlutterPlugin.stopAllMiniApp()；
```

Deleting the Mini Program

Last updated：2025-01-03 15:15:00

Note:

The mini program package and information is cached locally for faster opening of the mini program next time. If you want to delete all information of the mini program, you can call the following API to delete a specific mini program or all mini programs.

Delete a specific mini program

To delete all local data of a mini program with a given ID:

appId: Mini program ID

appVerType: Mini program type

version: Mini program version to be deleted

```
Future<void> deleteMiniAppCache(String appId,
   {int appVerType = 0, String? version})
```

appVerType

```
/// Constants for the mini program package types
class AppVerType {
  static const int online = 0;
  static const int develop = 1;
  static const int preview = 2;
  static const int experience = 3;
}
```

Example:

```
 _tcmppFlutterPlugin.deleteMiniAppCache(appId!,0,'1.0.0');
```

Delete all mini programs

```
Future<void> clearMiniAppCache()
```

Example:

```
 _tcmppFlutterPlugin.clearMiniAppCache();
```

Searching for the Mini Program

Last updated：2025-04-03 17:53:12

Search mini programs

keyWord: The keyword used to search the mini program.

searchAppType：the scope to be searched, refer to SearchAppType.

category： category information of the mini program.

pageInfo：specify the page index and size when searching.

```
Future<SearchMiniAppResult?> searchMiniApp(
    String keyword, {
      SearchAppType searchAppType = SearchAppType.searchAppTypeAll,
      CategoryInfo? category = const CategoryInfo(),
      PageInfo? pageInfo = const PageInfo()
})
```

Example:

```
Future<void> doSearch() async {
  FocusScope.of(context).unfocus();
  final result = await _tcmppFlutterPlugin
      .searchMiniApp(
        searchController.text,
        category: CategoryInfo(
          mainCategory: firstTypeController.text,
          subCategory: secondTypeController.text
        )
      )
      .catchError((err) {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text(err.toStr
      });

  setState(() {
    _appInfoList.clear();
    if (result != null && result.appInfoList.isNotEmpty) {
      _appInfoList.addAll(result.appInfoList);
      _emptyText = 'Total: ${result.total}, Current: ${result.appInfoList.length}';
    } else {
      _emptyText = 'No mini app found';
    }
  });
}
```

Get categories

Get all categories

existMnpOnly：Whether to get only the list of currently published mini programs.

searchAppType：the scope to be searched, refer to SearchAppType.

```
Future<List<CategoryInfo>?> getCategoryList(bool existMnpOnly,SearchAppType searchA
```

Example：

```
List<CategoryInfo>? ctList = await _tcmppFlutterPlugin.getCategoryList(true, Search
if (ctList != null) {
  for (var i = 0; i < ctList.length; i++) {
    print('Main Category: ${ctList[i].mainCategory}');
    print('Sub Category: ${ctList[i].subCategory}');
  }
}
```

Get a list of recent visits to the mini program

Last updated：2024-08-06 15:52:10

The tcmpp_flutter plugin provides the following API for accessing the list of recently used mini programs.

```
Future<List<AppInfo>?> getRecentList()
```

Sample code:

```
_tcmppFlutterPlugin.getRecentList();
```

Pre-downloading a Mini Program

Last updated：2024-08-06 15:51:49

Since the mini program checks for updates and synchronizes versions upon opening, there may be a waiting period. To minimize the startup time and enhance the user experience, the following API is provided to pre-download the mini program packages.

appIdList: A list of mini program IDs to be loaded.

isDownload: If set to "true", the mini program package will be downloaded after preloading. The default value is "false".

```
Future<void> preDownloadMiniApp(List<String> appIdList,
   {bool isDownload = false})
```

Sample code:

```
List<String> appIdList = ['app_id_1', 'app_id_2', 'app_id_3'];
_tcmppFlutterPlugin.preDownloadMiniApp(appIdList, isDownload: true);
```

Mini Program Capabilities Customization
Customizing Mini Program APIs

Last updated：2024-09-10 18:52:52

If a mini program needs capabilities from the host app that the SDK doesn't support, developers can register custom APIs to enable these capabilities. This allows the mini program to call custom APIs provided by the host application. Define a custom mini program API and associate it with an API handler. apiName: The name of the custom API. The mini program can use this API by calling wx.invokeNativePlugin().
apiHandler: The function that handles the API calls.

```
void registerMiniAppApi(String apiName, TcmppMiniAppApiHandler apiHandler)
```

Sample code:

```
final _tcmppFlutterPlugin = TcmppFlutter();

...

@override
void initState() {
   super.initState();

   ...

   /// The custom APIs must be registered prior to mini program startup
   _tcmppFlutterPlugin.registerMiniAppApi("myApiName",myApiHandler);
}
```

The mini program uses wx.invokeNativePlugin with api_name to call the Flutter-registered myApiName.

Sample code:

```
example code:
   /// Mini Program Call
 var opts = {
 api_name: 'myApiName',
 success: function(res) {
   log(res);
 },
 fail: function(res) {
   log(res);
 },
 complete: function(res) {
   log(res);
```

```
  },
  data: {
      name : 'kka',
      age : 22
   }
  }
  wx.invokeNativePlugin(opts);
```

Implement the myApiHandler function on the Flutter side:

```
/// client API
Future<Map<String, dynamic>?> myApiHandler(MiniApiCall call) async {
    print("API : ${call.apiName}");
    print("AppInfo: ${call.appInfo}");
    print("WebView ID: ${call.webViewId}");
    print("params: ${call.params}");


    return {"result": "success","method":"myApiHandler"};
}
```

Plugin Customization
Customizing Plugin Capabilities

Last updated：2024-08-06 15:44:43

You can associate the data interaction between the mini program and the host app by implementing the OpenApiHandler and TcmppPlatformEventHandler abstract classes.

Sample code:

```
_tcmppFlutterPlugin.registerOpenApiHandler(MyOpenApiHandler());
_tcmppFlutterPlugin.registerPlatformEventHandler(MyPlatformHandler());
```

Sample code for API class implementation:

```
class MyPlatformHandler extends TcmppPlatformEventHandler {


  @override
  Future<List<CustomMenu>> getCustomMenus() async {
    CustomMenu menu1 = CustomMenu(
        '100', 'res/images/mini_app_wechat_friend.png', 'Share To', true,
        shareKey: 'twitter');
    CustomMenu menu2 = CustomMenu(
        '101',
        'https://staticintl.cloudcachetci.com/cms/backend-cms/8WGP653_%E5%BC%80%E5%
        'Custom',
        false);


    return [
      menu1,
      menu2,
    ];
  }


  @override
  Future<void> customMenuClick(String menuId, ShareData? shareMenu) async {
    print("click menuId:$menuId shareMenu:$shareMenu");
    throw UnimplementedError();
  }


  @override
  Future<bool> reportEvent(int eventId, String eventName, AppInfo appInfo,
```

```
      Map<Object?, Object?> params) async {
    print("reportEvent:$eventName  appinfo:$appInfo params:$params");
    // TODO: implement reportEvent
    return true;
  }


  @override
  Future<void> onMiniProgramStateChange(
      String appId, MiniProgramState state) async {
    print("app state change: appid=$appId, state=$state");
  }
}
```

Customizing Mini Program Information API

Last updated：2024-08-06 15:36:18

Monitoring mini program lifecycle

The Flutter app host can receive notifications by overriding the onMiniProgramStateChange method when the mini program starts, enters the foreground, background, or closes.
Sample code:

```
@override
Future<void> onMiniProgramStateChange(
    String appId, MiniProgramState state) async {
  print("app state change: appid=$appId, state=$state");
}
```

Customizing the Sharing Capability

Last updated：2024-08-06 15:35:33

Developers can customize the buttons on the operation panel of the mini program, including the sharing button and other buttons. The operation panel can be brought up by clicking the More button in the top right corner of the mini program:

```
@override
Future<List<CustomMenu>> getCustomMenus() async {
/// The menu contains menuId, image (local paths and network images are supported),
  CustomMenu menu1 = CustomMenu(
      '100', 'res/images/mini_app_wechat_friend.png', 'Share To', true,
      shareKey: 'twitter');
  CustomMenu menu2 = CustomMenu(
      '101', 'https://staticintl.cloudcachetci.com/cms/backend-cms/8WGP653_%E5%BC%8


  return [
    menu1,
    menu2,
  ];
}
```

Menu click callback:

```
@override
Future<void> customMenuClick(String menuId, ShareData? shareMenu) async {
  /// Callback for menu button click
  print("click menuId:$menuId shareMenu:$shareMenu");
  throw UnimplementedError();
}
```

The menu has built-in buttons for four sharing channels: QQ, Qzone, WeChat, and WeChat Circle of Friends. If you want to display these sharing buttons, you can configure them in the following ways.

```
_tcmppFlutterPlugin.defaultShareTargets([ShareTarget.qq, ShareTarget.wxMoment]);
```

Customizing User Attributes

Last updated：2024-08-06 15:32:03

Set the current login account for data isolation and user information display.

The account must be set before mini program startup.

info: The information of the current login account. It is left empty if no user is logged in.

```
void setAccount(AccountInfo? info)
```

Sample code:

```
_tcmppFlutterPlugin.setAccount(AccountInfo(
        uid: "12345",
        avatarUrl: "https://picsum.photos/250?image=9",
        accountName: "SimpleAccount"));
```

Event Reporting

Last updated：2024-08-06 15:24:26

The event reporting API can be implemented in the host app to override the reporting logic in tcmpp_flutter.

```
Future<bool> reportEvent(int eventId, String eventName, AppInfo appInfo,
      Map<Object?, Object?> params) async {
    return false;
  }
```

Sample code:

```
@override
  Future<bool> reportEvent(int eventId, String eventName, AppInfo appInfo,
      Map<Object?, Object?> params) async {
    print("reportEvent:$eventName  appinfo:$appInfo params:$params");
    // TODO: implement reportEvent
    return true;
  }
```

Open APIs

Last updated：2024-08-06 15:23:50

The mini program SDK provides some open APIs for implementing capabilities such as login, obtaining user information, and payment, which are provided by host apps. See the following table for the supported open APIs:

Mini program	MiniOpenApiProxy	Description
wx.login	login	Login API
wx.getUserInfo	getUserInfo	Obtain basic user information
wx.getUserProfile	getUserProfile	Obtain user attribute information
wx.getPhoneNumber	getPhoneNumber	Obtain phone number
wx.requestPayment	requestPayment	Initiate a payment
wx.checkSession	checkSession	Check if the login has expired

You can associate data interactions between the mini program and the host application by implementing the OpenApiHandler abstract class

```
abstract class OpenApiHandler {
  // The API is called when the mini program calls wx.requestPayment to request a t

  Future<Map<String, dynamic>> requestPayment(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// The API is called when the mini program calls wx.getUserProfile to request th
  ///
  Future<Map<String, dynamic>> getUserProfile(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// The API is called when the mini program calls wx.login to request the login c
  ///
  Future<Map<String, dynamic>> login(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// The API is called when the mini program calls wx.checkSession to request the
  /// Check if the login has expired
  ///
  Future<Map<String, dynamic>> checkSession(
      AppInfo appInfo, Map<Object?, Object?> params);
```

```
  /// The API is called when the mini program calls wx.getUserProfile
  /// Compatible with earlier mini program APIs
  ///
  Future<Map<String, dynamic>> getUserInfo(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// The API is called when the mini program calls wx.getPhoneNumber to request th
  ///
  Future<Map<String, dynamic>> getPhoneNumber(
      AppInfo appInfo, Map<Object?, Object?> params);
}
```

Sample code for obtaining user information instance:

```
@override
  Future<Map<String, dynamic>> getUserProfile(
      AppInfo appInfo, Map<Object?, Object?> params) async {
    print("getUserProfile:$appInfo params:$params");
    Map<String, dynamic> result = {
      "userInfo": {
        "nickName": "xcode",
        "avatarUrl":
            "https://staticintl.cloudcachetci.com/cms/backend-cms/8WGP653_%E5%BC%80
        "gender": 1,
        "country": "China",
        "province": "ChongQing",
        "city": "ChongQing",
      }
    };
    return Future.value(result);
  }
```

Others

Last updated：2024-08-06 15:21:02

Setting the topic for the mini program container

The topic must be set before mini program startup. You can choose light mode, dark mode, or using default settings.

```
Future<void> setTheme(MiniTheme theme)
```

Sample code:

```
_tcmppFlutterPlugin.setTheme(MiniTheme.dark);
```

Localization setting for the mini program

It must be set before the mini program startup.

language: The language used in the mini program. The ISO 639 alpha-2 or alpha-3 language code should be used.

The ISO 3166 alpha-2 country code should be used.

variant: An arbitrary value used to represent a variant of the local language.

```
Future<void> setLocale(String language,
      {String? region, String? variant})
```

Sample code:

```
_tcmppFlutterPlugin.setLocale("en", region: "us");
```

API Description

Last updated：2024-08-06 15:17:13

MiniStartOptions

```
class MiniStartOptions {
  /// Entry path to the mini program.
  String? entryPath;

  /// Always update the mini program on startup.
  bool? isForceUpdate;

  /// String parameters to pass to the mini program on
  String? params;
}
```

ScanResult

```
class ScanResult {
  /// The result string for a qrcode or barcode contains
  String? result;

  /// code type
  String? scanType;

  /// Character encoding of the result string
  String? charset;
}
```

AppInfo

```
class AppInfo {
  /// Mini program id.
  String appId;

  /// The mini program package type (distribution, development, etc.). See [AppVerT
```

```
  int appVerType;

  /// The mini program version.
  String version;

  /// Name of the mini program.
  String? name;

  /// The mini program icon link.
  String? iconUrl;

  /// Description of the mini program.
  String? appIntro;

  /// Mini program developer.
  String? appDeveloper;

  /// Mini program release time.
  int time;
}
```

AppVerType

```
/// Constants for mini program package types, see [AppInfo].
class AppVerType {
  static const int online = 0;
  static const int develop = 1;
  static const int preview = 2;
  static const int experience = 3;
}
```

AccountInfo

```
class AccountInfo {
  /// Unique ID of the current account
  String? uid;

  /// Link to the avatar
  String? avatarUrl;
```

```
  /// Current account name
  String? accountName;
}
```

OpenApiHandler

```
abstract class OpenApiHandler {
  // Called when the mini program calls wx.requestPayment to request a third-party
  Future<Map<String, dynamic>> requestPayment(
      AppInfo appInfo, Map<Object?, Object?> params);

  //// Called when the mini program calls wx.getUserProfile to request user informa
  ///
  Future<Map<String, dynamic>> getUserProfile(
      AppInfo appInfo, Map<Object?, Object?> params);

   /// Called when the mini program calls wx.login, requesting the host application
  ///
  Future<Map<String, dynamic>> login(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// Called when the mini program calls wx.checkSession, requesting the host appli
  /// Checks if the login has expired
  ///
  Future<Map<String, dynamic>> checkSession(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// Called when the mini program calls wx.getUserInfo, which has been outdated by
  /// Compatible with earlier mini program api.
  ///
  Future<Map<String, dynamic>> getUserInfo(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// Called when the mini program calls wx.getPhoneNumber to get the current user'
  ///
  Future<Map<String, dynamic>> getPhoneNumber(
      AppInfo appInfo, Map<Object?, Object?> params);
}
```

TcmppPlatformEventHandler

```
abstract class TcmppPlatformEventHandler {


  Future<String> getAppName() async {
    return "";
  }

  Future<String> getAppVersion() async {
    return "";
  }

  Future<List<CustomMenu>> getCustomMenus() async {
    return [];
  }


  Future<void> customMenuClick(String menuId, ShareData? shareMenu) async {}

  Future<void> onMiniProgramStateChange(
      String appId, MiniProgramState state) async {}


  Future<bool> reportEvent(int eventId, String eventName, AppInfo appInfo,
      Map<Object?, Object?> params) async {
    return false;
  }
}
```

Extensions

Last updated：2025-01-16 19:17:49

Many APIs that require system authorization for development and use need to be pre-authorized in the info.plistfile of iOS and the AndroidManifest.xmlfile of Android. However, your app may not need this feature, so tcmpp_flutter splits out the extension SDK, which eliminates unnecessary authorization and reduces the size of the core module. tcmpp_flutter provides the core module and the expansion module for users to access as needed.
Some mini program APIs may require additional privacy and permissions. To use these APIs, additional Flutter plugin dependencies are required.

APIs	Plugin names	Required permissions
LBS-related APIs (location and POI search)	tcmpp_flutter_lbs	Access location
MDNS APIs	tcmpp_flutter_mdns	Access local network
TCP/UDP APIs	tcmpp_flutter_network	Access network
Media APIs (images & videos)	tcmpp_flutter_media	Access image library
Wireless API, bluetooth API, calendar API, contacts API, clipboard API, biometric authentication API	tcmpp_flutter_device	Wireless API, bluetooth API, calendar API, contacts API, clipboard API, biometric authentication API
Map APIs	tcmpp_flutter_googlemap (for general) tcmpp_flutter_petalmap (for Huawei device)	Access location

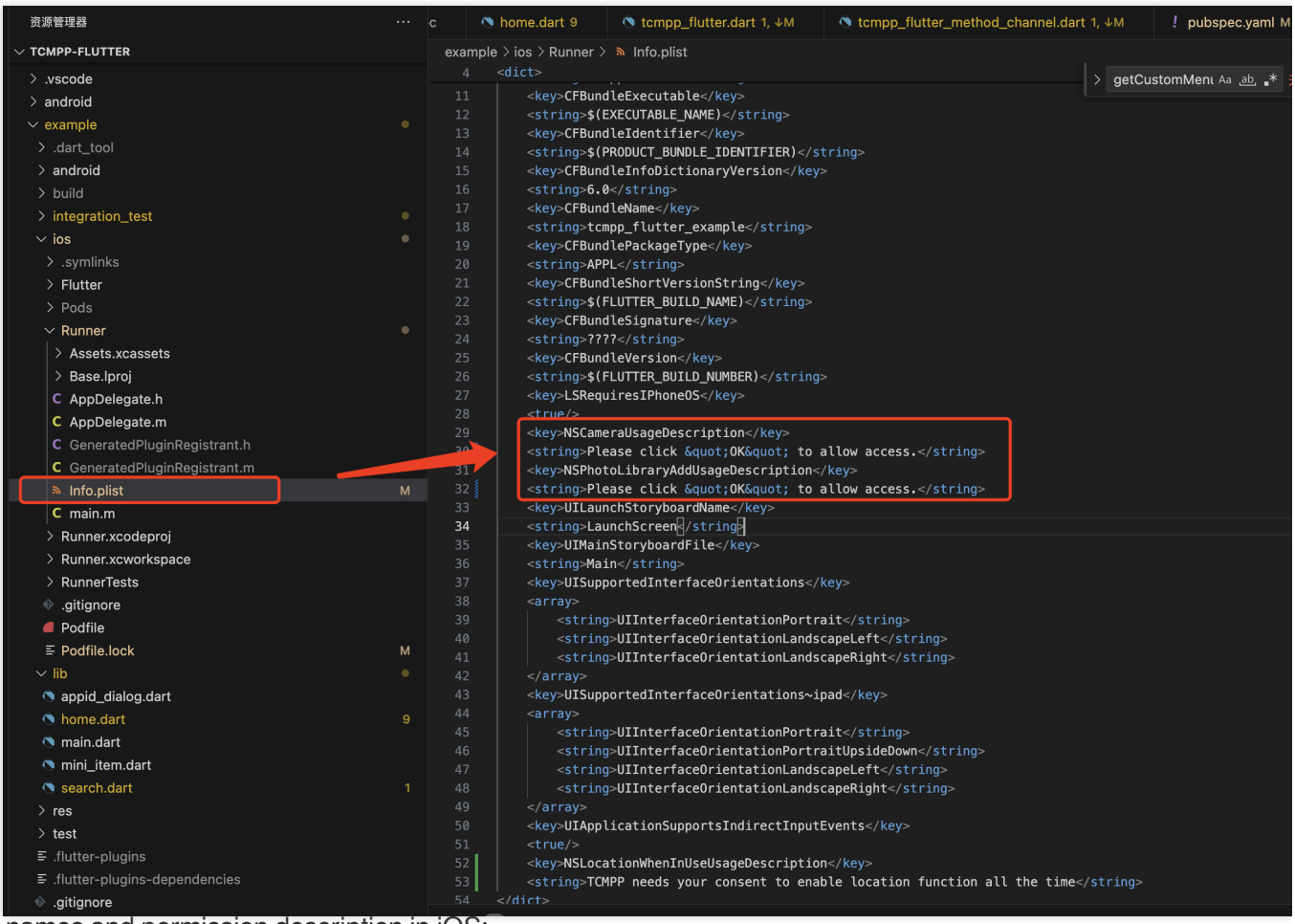
These plugins must be used together with tcmpp_flutter plugins. Map APIs are required only in Android, and they are included in iOS by default.

Flutter Privacy Compliance

Last updated：2024-08-06 15:18:18

Note：

Permission descriptions should be provided in the plist file in iOS.



Plugin names and permission description in iOS:

Plugin name	System permissions
tcmpp_flutter_lbs	NSLocationAlwaysUsageDescription NSLocationWhenInUseUsageDescription NSLocationAlwaysAndWhenInUseUsageDescription NSLocationUsageDescription NSMotionUsageDescription
tcmpp_flutter_mdns	NSBonjourServices
tcmpp_flutter_network	NSAppTransportSecurity
tcmpp_flutter_media/tcmpp_flutter	NSPhotoLibraryUsageDescription

	NSCameraUsageDescription
tcmpp_flutter_device	NSCalendarsUsageDescription NSRemindersUsageDescription NSContactsUsageDescription NSFaceIDUsageDescription NSBluetoothAlwaysUsageDescription NSBluetoothPeripheralUsageDescription

Note：

Permission descriptions should be provided in the android/app/src/main/AndroidManifest.xml file in Android.

Plugin names and permission description in Android:

Plugin name	System permissions
tcmpp_flutter	android.permission.CAMERA android.permission.WRITE_EXTERNAL_STORAGE android.permission.FLASHLIGHT
tcmpp_flutter_lbs	android.permission.ACCESS_FINE_LOCATION
tcmpp_flutter_network	android.permission.INTERNET android.permission.ACCESS_NETWORK_STATE
tcmpp_flutter_media	none
tcmpp_flutter_device	android.permission.BLUETOOTH_ADMIN android.permission.BLUETOOTH_SCAN android.permission.BLUETOOTH_ADVERTISE android.permission.BLUETOOTH_CONNECT android.permission.WRITE_CONTACTS android.permission.READ_CONTACTS android.permission.USE_FINGERPRINT android.permission.USE_BIOMETRIC android.permission.READ_CALENDAR android.permission.WRITE_CALENDAR android.permission.ACCESS_FINE_LOCATION android.permission.ACCESS_WIFI_STATE android.permission.CHANGE_WIFI_STATE android.permission.ACCESS_NETWORK_STATE android.permission.CHANGE_NETWORK_STATE
tcmpp_flutter_googlemap	android.permission.ACCESS_FINE_LOCATION
tcmpp_flutter_petalmap	android.permission.INTERNET android.permission.ACCESS_NETWORK_STATE

	android.permission.CHANGE_WIFI_STATE android.permission.ACCESS_COARSE_LOCATION android.permission.ACCESS_FINE_LOCATION

Flutter Plugin

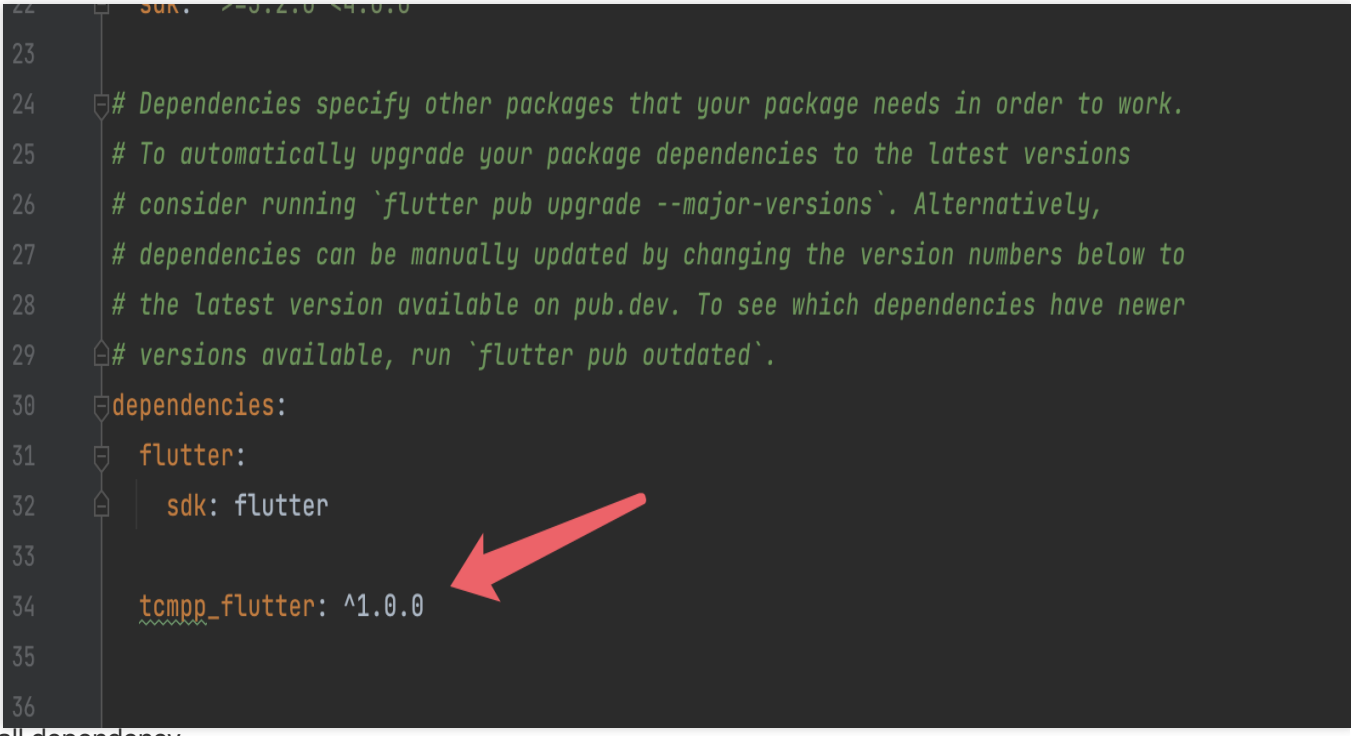
Last updated：2025-02-10 17:12:53

Flutter Plugin is a Flutter plugin to provide access for SDK in Flutter.

Add Flutter Plugin

Add the core plugin from Pub.dev

1. Add dependencyTo add plugin from pub.dev, open the pubspec.yaml file located inside the app folder, and add tcmpp_flutter: ${version} under dependencies.



2. Install dependency

From the terminal: Run flutter pub get.

From VS Code: Click Get Packages located in right side of the action ribbon at the top of pubspec.yaml indicated by the Download icon.

From Android Studio/IntelliJ: Click Pub get in the action ribbon at the top of pubspec.yaml.

For details how to add Flutter plugins or packages, see Use packages & plugins.

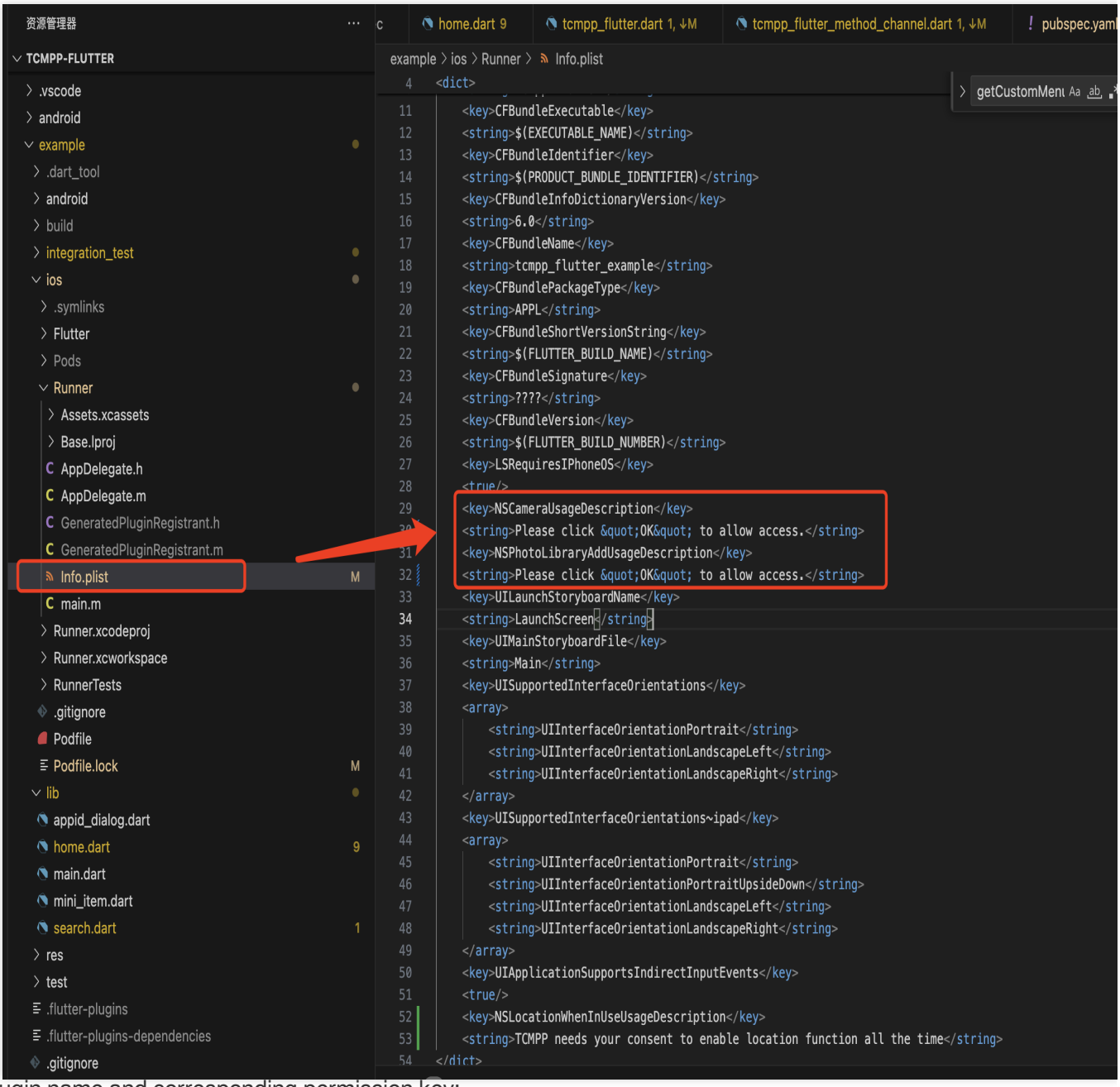
Add the extra plugins

Some mini-program APIs may require additional privacy and permissions. To use these APIs, additional Flutter plugin dependencies are needed.

APIs	Plugin name	Permissions needed
LBS related APIs (Location & POI search)	tcmpp_flutter_lbs	Access location
MDNS APIs	tcmpp_flutter_mdns	Access locale network
TCP/UDP APIs	tcmpp_flutter_network	Access internet
Meida APIs (Photos & Videos)	tcmpp_flutter_media	Access photo library
Wifi APIs, Bluetooth APIs, Calendar APIs, Contact APIs, Clipboard APIs, Biometric authentication APIs	tcmpp_flutter_device	Access wifi state, Access bluetooth, Access Calendar, Access contact, Access clipboard, Access fingerprint & face id
Map APIs	tcmpp_flutter_googlemap (for general) tcmpp_flutter_petalmap (for Huawei device)	Access location

These plugins must be used together with ***tcmpp_flutter*** plugin. Map API support is only needed by Android platform, iOS include these APIs by default.

[Notice] iOS needs to add corresponding permission descriptions in the ***plist*** file：



iOS plugin name and corresponding permission key:

Plugin name	Permission key
tcmpp_flutter_lbs	NSLocationAlwaysUsageDescription NSLocationWhenInUseUsageDescription NSLocationAlwaysAndWhenInUseUsageDescription NSLocationUsageDescription NSMotionUsageDescription
tcmpp_flutter_mdns	NSBonjourServices
tcmpp_flutter_network	NSAppTransportSecurity
tcmpp_flutter_media	NSPhotoLibraryUsageDescription

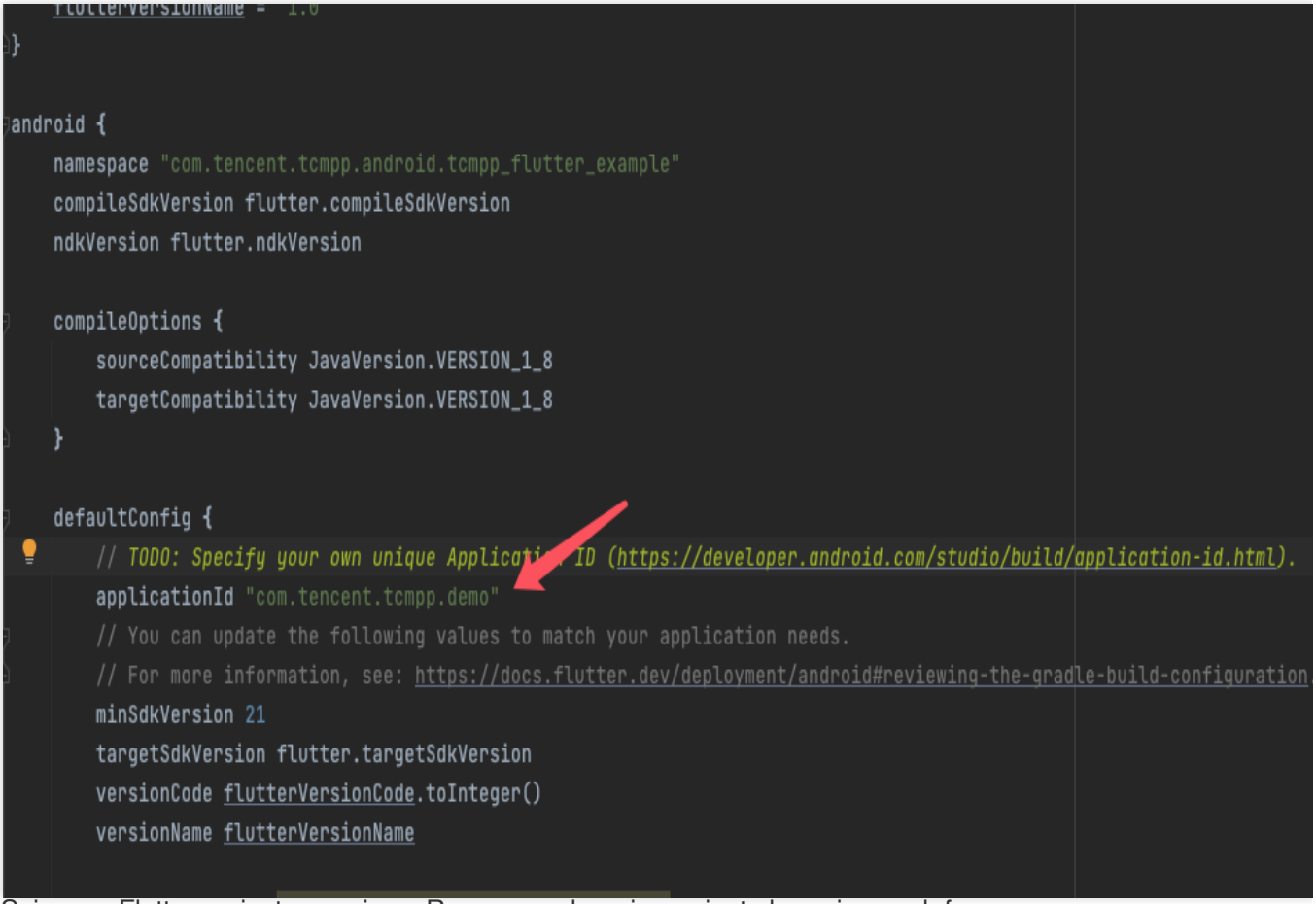
	NSCameraUsageDescription
tcmpp_flutter_device	NSCalendarsUsageDescription NSRemindersUsageDescription NSContactsUsageDescription NSFaceIDUsageDescription NSBluetoothAlwaysUsageDescription NSBluetoothPeripheralUsageDescription

Add assets for the configuration files

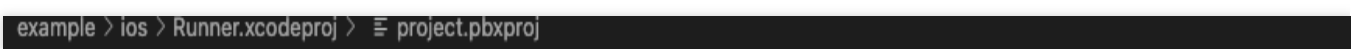
Get the configuration file from the product console

1. Get application ID / bundle ID of your Android / IOS application.

For Android, in your Flutter project, open android > app > build.gradle, application id can be found in android > defaultConfig section.



For IOS, in your Flutter project, open ios > Runner.xcodeproj > project.pbxproj, search for PRODUCT_BUNDLE_IDENTIFIER.



```
424  /* Begin XCBuildConfiguration section */
645          };
646          97C147061CF9000F007C117D /* Debug */ = {
647              isa = XCBuildConfiguration;
648              baseConfigurationReference = 9740EEB21CF90195004384FC /* Debug.xcconfig */;
649              buildSettings = {
650                  ASSETCATALOG_COMPILER_APPICON_NAME = AppIcon;
651                  CODE_SIGN_STYLE = Manual;
652                  CURRENT_PROJECT_VERSION = "$(FLUTTER_BUILD_NUMBER)";
653                  DEVELOPMENT_TEAM = "";
654                  "DEVELOPMENT_TEAM[sdk=iphoneos*]" = VU8Y4N38VF;
655                  ENABLE_BITCODE = NO;
656                  "EXCLUDED_ARCHS[sdk=iphonesimulator*]" = "i386 arm64";
657                  INFOPLIST_FILE = Runner/Info.plist;
658                  LD_RUNPATH_SEARCH_PATHS = (
659                      "$(inherited)",
660                      "@executable_path/Frameworks",
661                  );
662                  PRODUCT_BUNDLE_IDENTIFIER = com.tencent.tcmpp.demo;
663                  PRODUCT_NAME = "$(TARGET_NAME)";
664                  PROVISIONING_PROFILE_SPECIFIER = "";
665                  "PROVISIONING_PROFILE_SPECIFIER[sdk=iphoneos*]" = tcmpp_demo;
666                  VERSIONING_SYSTEM = "apple-generic";
667              };
668              name = Debug;
669          };
670          97C147071CF9000F007C117D /* Release */ = {
671              isa = XCBuildConfiguration;
672              baseConfigurationReference = 7AFA3C8E1D35360C0083082E /* Release.xcconfig */;
673              buildSettings = {
674                  ASSETCATALOG_COMPILER_APPICON_NAME = AppIcon;
675                  CODE_SIGN_STYLE = Manual;
676                  CURRENT_PROJECT_VERSION = "$(FLUTTER_BUILD_NUMBER)";
677                  DEVELOPMENT_TEAM = "";
678                  "DEVELOPMENT_TEAM[sdk=iphoneos*]" = VU8Y4N38VF;
679                  ENABLE_BITCODE = NO;
680                  "EXCLUDED_ARCHS[sdk=iphonesimulator*]" = "i386 arm64";
681                  INFOPLIST_FILE = Runner/Info.plist;
682                  LD_RUNPATH_SEARCH_PATHS = (
683                      "$(inherited)",
684                      "@executable_path/Frameworks",
685                  );
686                  PRODUCT_BUNDLE_IDENTIFIER = com.tencent.tcmpp.demo;
687                  PRODUCT_NAME = "$(TARGET_NAME)";
688                  PROVISIONING_PROFILE_SPECIFIER = "";
689                  "PROVISIONING_PROFILE_SPECIFIER[sdk=iphoneos*]" = tcmpp_demo;
690                  VERSIONING_SYSTEM = "apple-generic";
691              };
692              name = Release;
693          };
```

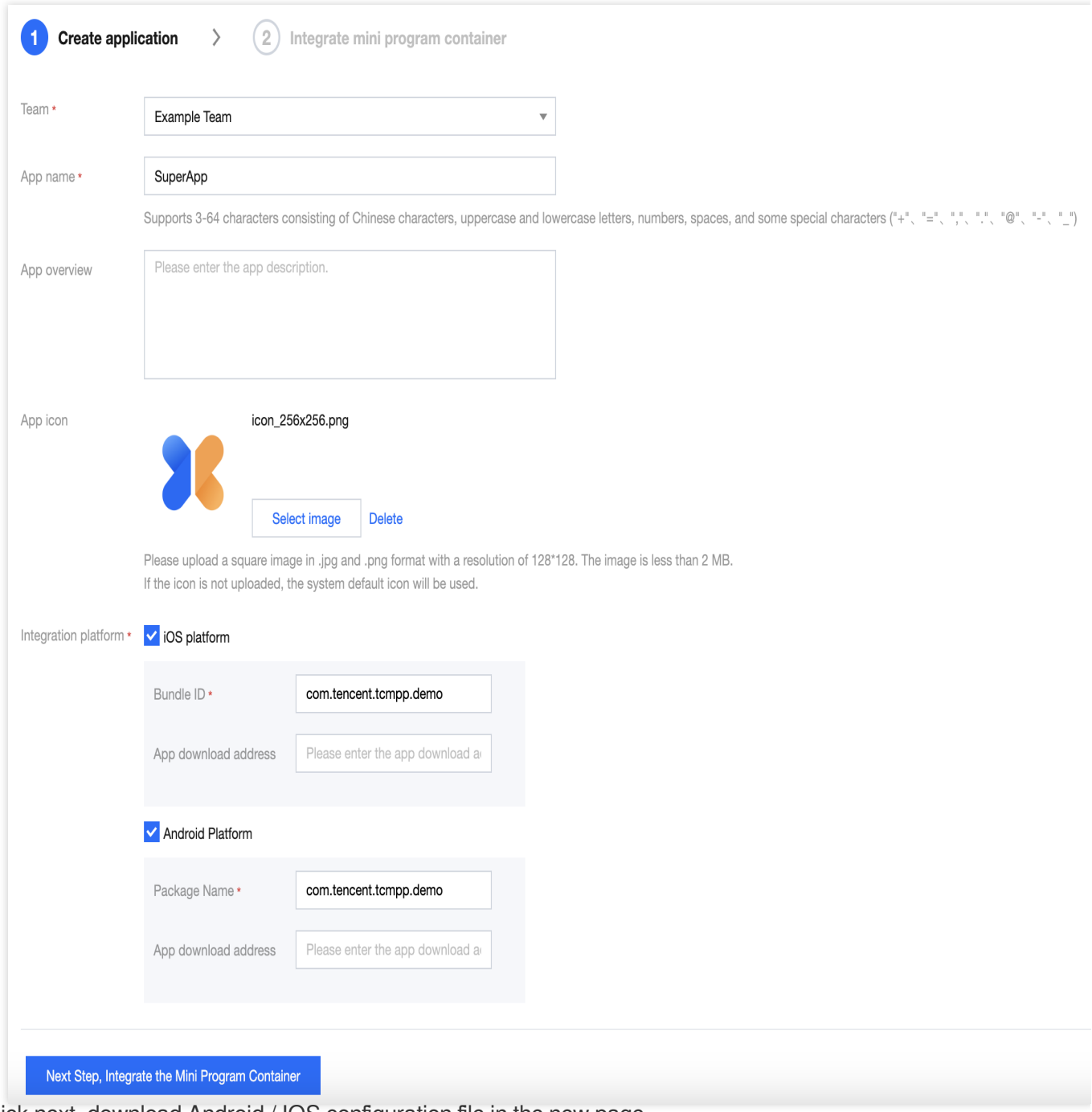
2. Create Flutter Application on the web console.

2.1 Login the Tencent Cloud Mini Program Platform Console and click 'Overview' in the left-hand navigation bar.

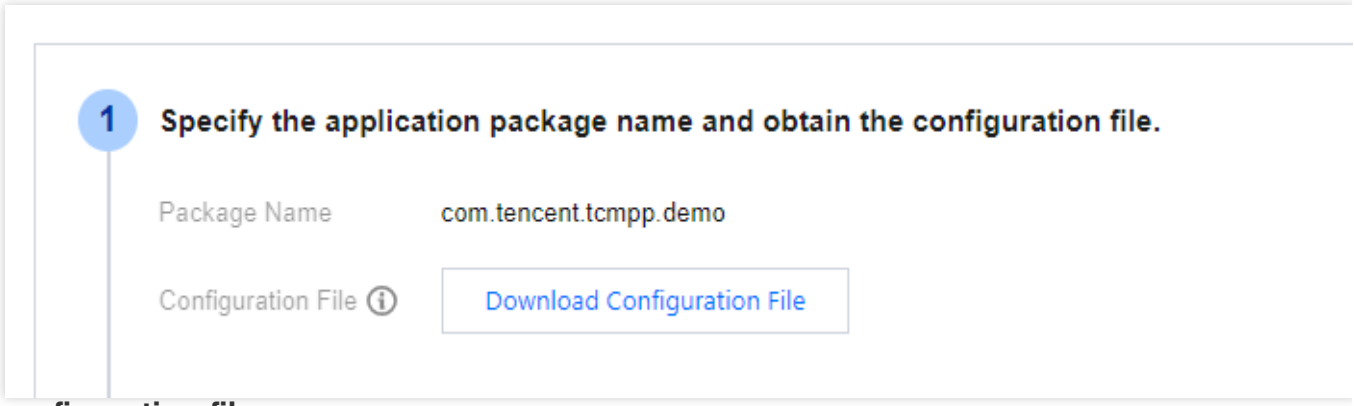
2.2 On the Overview page, click **Access Application**.

2.3 In the pop-up window for creating an application, fill in informations for your application.

2.4 For Intergration platform section, check both IOS platfrom & Android Platform. Then fill application ID / bundle ID you got in the package name field.

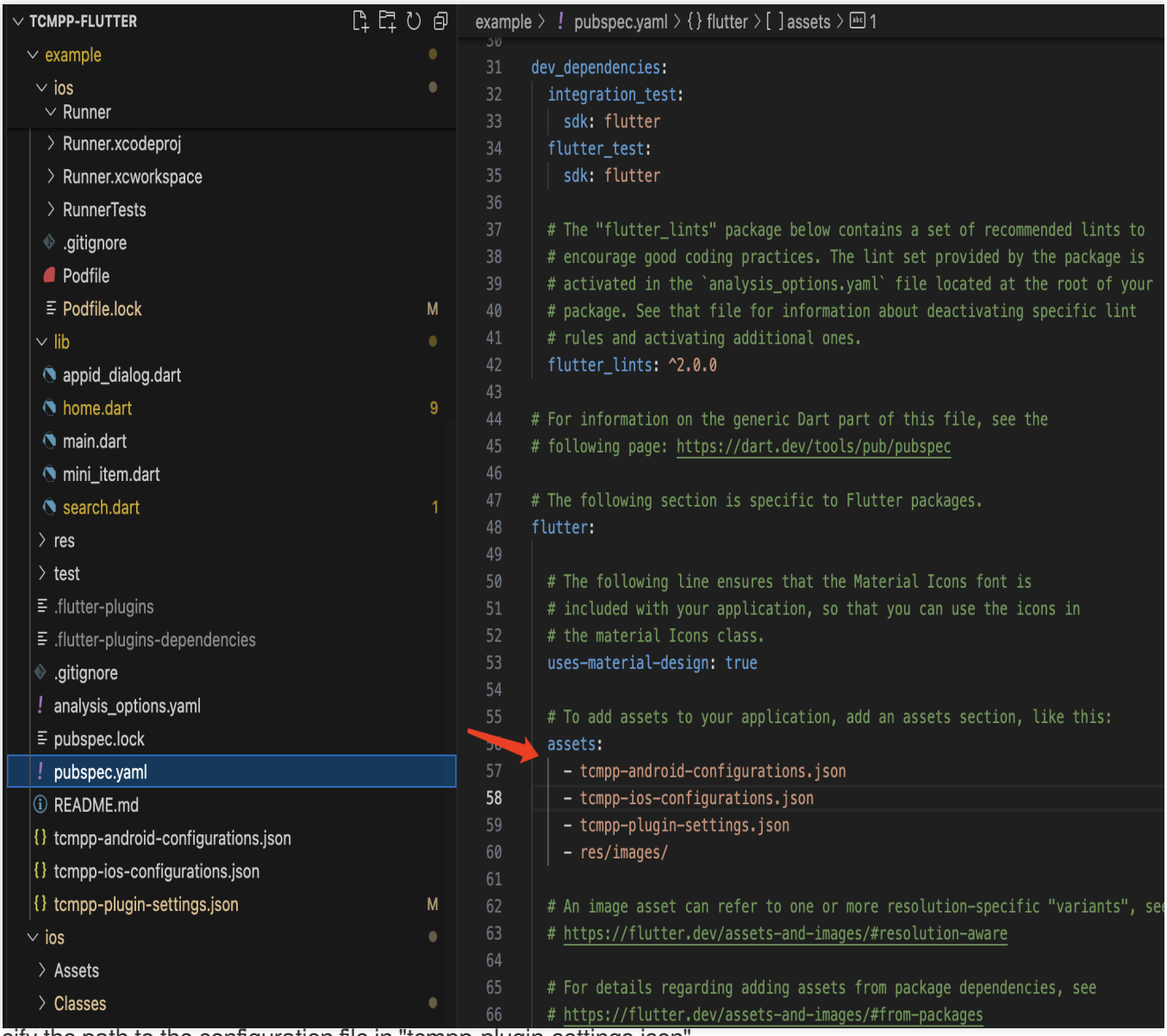


2.5 Click next, download Android / IOS configuration file in the new page.

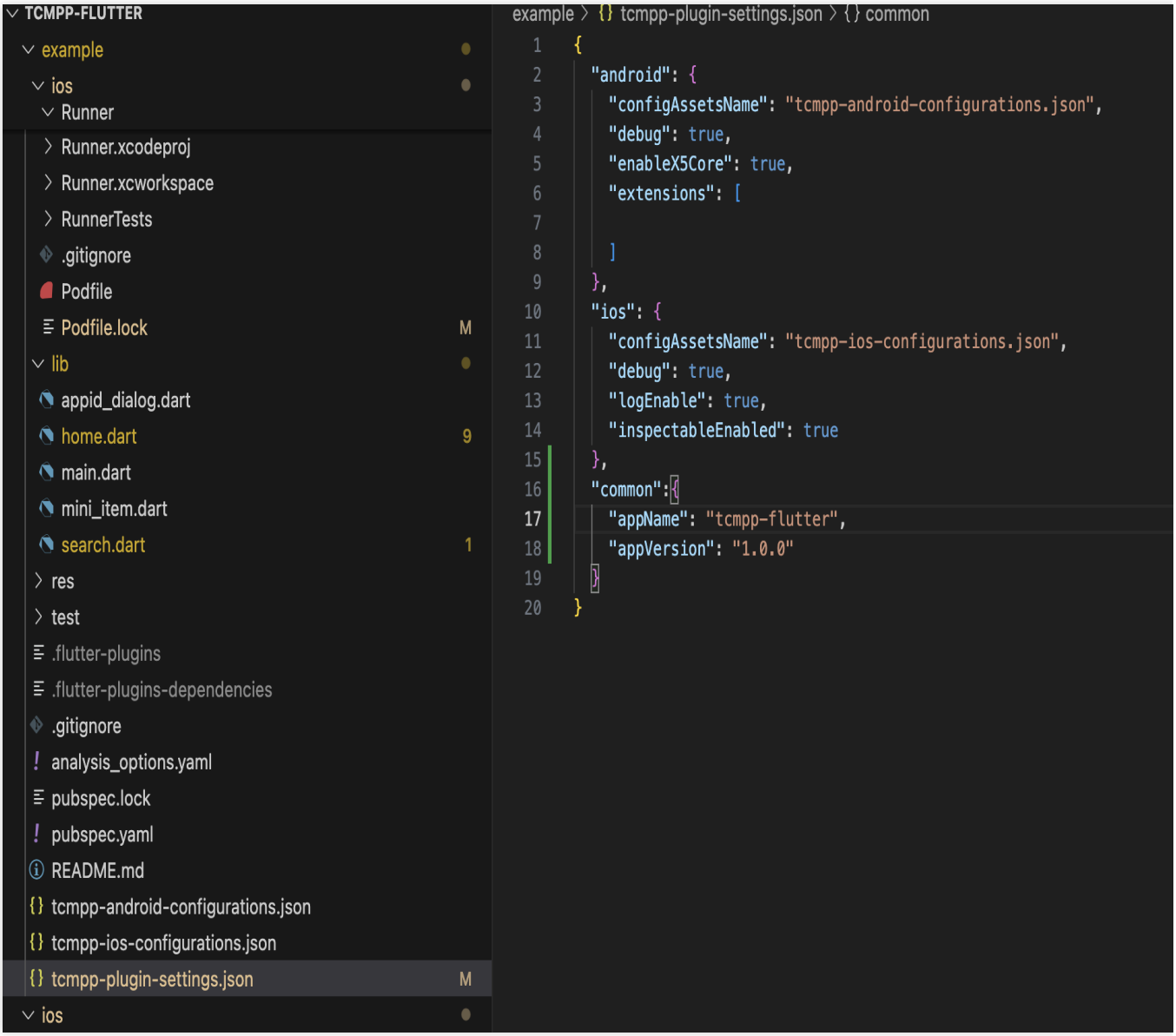


Add configuration file

1. Put the configuration file someware in your flutter project.
2. Add a file named "tcmpp-plugin-settings.json" in your project's root.
3. Edit **pubspec.yaml** file and add flutter assets for files mentioned above.



4. Specify the path to the configuration file in "tcmpp-plugin-settings.json".



SDK settings

In the **tcmpp-plugin-settings.json** file, you can choose to configure settings of your SDK.

For Android:

Field name	Type	Description
configAssetsName	string	Android configuration file path
debug	bool	enable SDK debugging & log
enableX5Core	bool	Whether to use TBS service(X5 core) as mini-program runtime
x5LicenseKey	string	License for TBS service if runtime enabled

X5DocLicenseKey	string	License for TBS document service if runtime enabled
x5Core32Url	string	Download url for 32-bit TBS runtime
x5Core64Url	string	Download url for 64-bit TBS runtime

For iOS:

Field name	Type	Description
configAssetsName	string	iOS configuration file path
debug	bool	enable SDK debug
logEnable	bool	enable SDK log
inspectableEnabled	bool	Enable inspectable after iOS 16.4, you can debug mini-program through safari

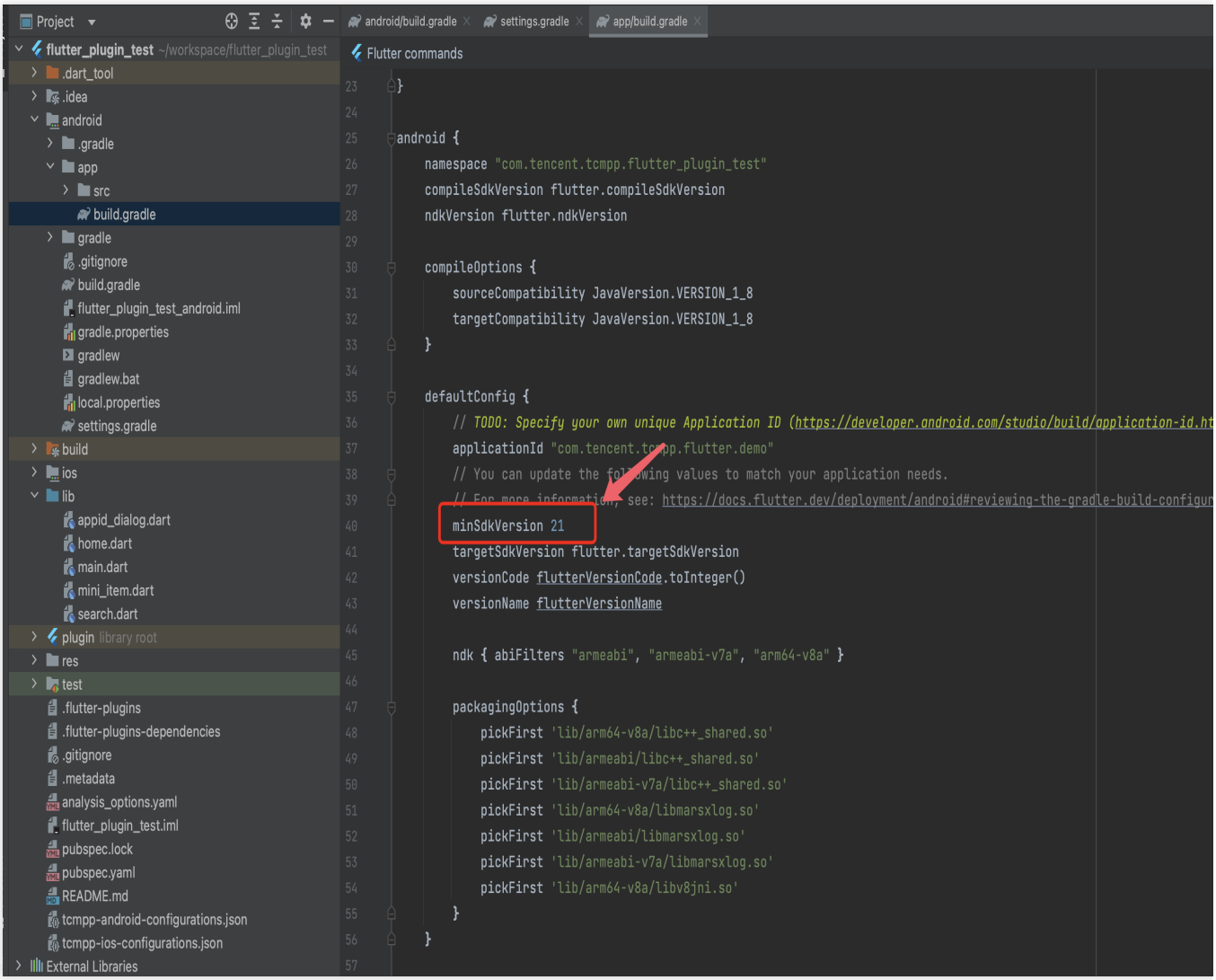
For common:

Field name	Type	Description
appName	string	Host app name, mainly used for copyright prompts in mini-program
appVersion	string	Host app version, mainly used for copyright prompts in mini-program

Platform specific

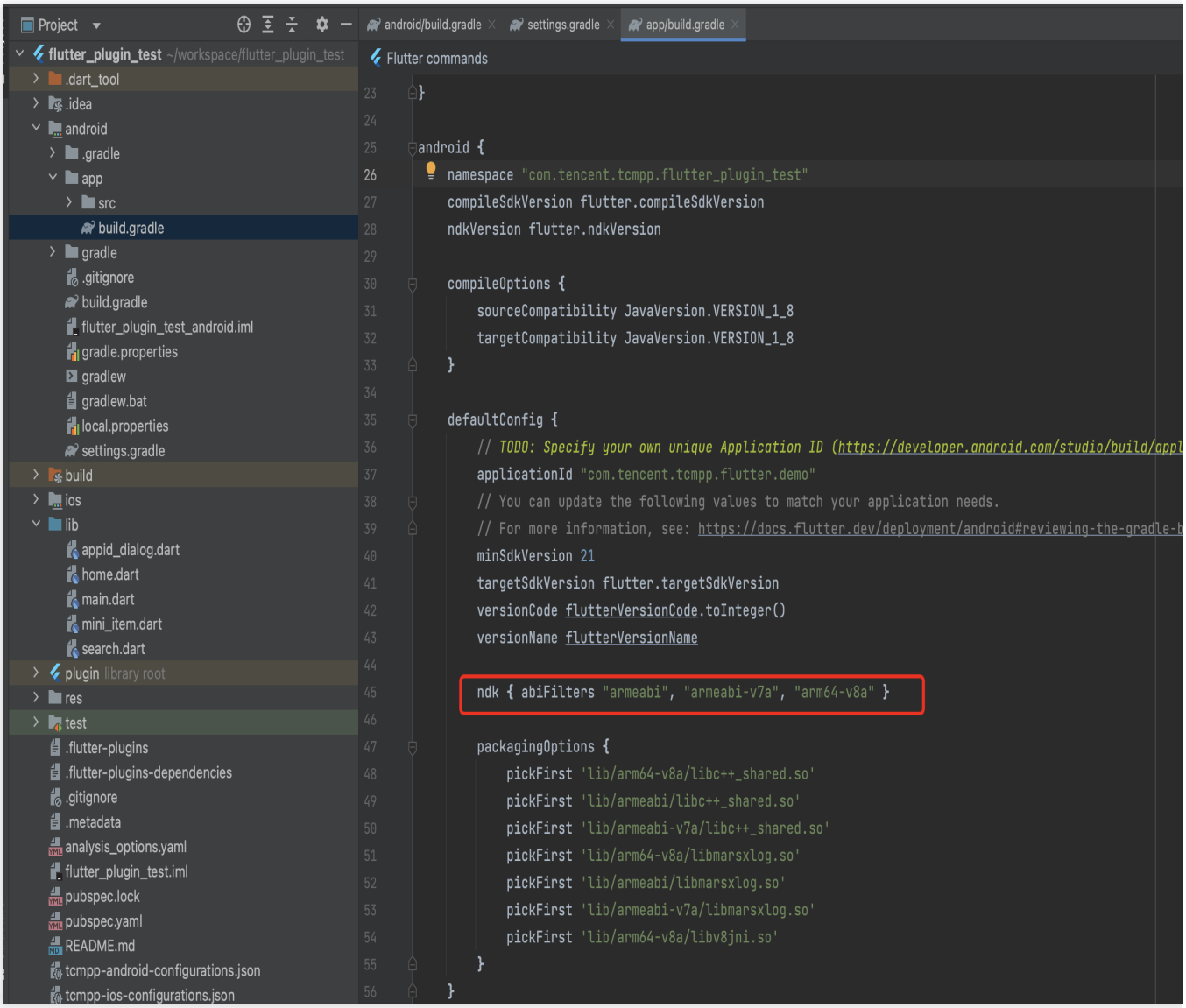
For Android platform

1. Open android > app directory, edit build.gradle file.
2. In android > defaultConfig section, change minSdkVersion to no less than 21.



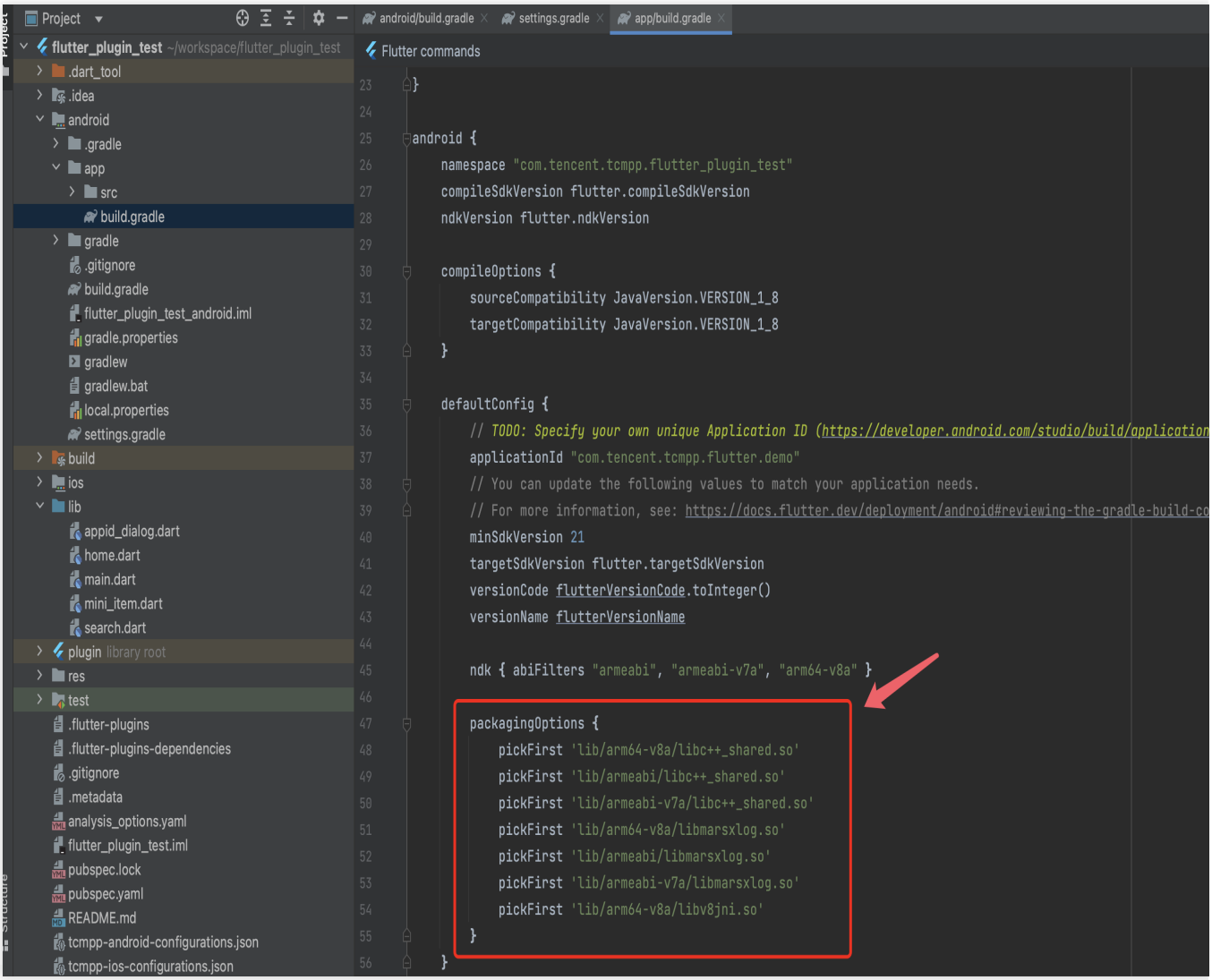

```
minSdkVersion 21
```

3. In android > defaultConfig section, add ndk filter for **armeabi**, **armeabi-v7a**, **arm64-v8a**.




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

4. In android > defaultConfig section, add packagingOptions.




```
packagingOptions {
    pickFirst 'lib/arm64-v8a/libc++_shared.so'
    pickFirst 'lib/armeabi/libc++_shared.so'
    pickFirst 'lib/armeabi-v7a/libc++_shared.so'
    pickFirst 'lib/arm64-v8a/libmarsxlog.so'
    pickFirst 'lib/armeabi/libmarsxlog.so'
    pickFirst 'lib/armeabi-v7a/libmarsxlog.so'
    pickFirst 'lib/arm64-v8a/libv8jni.so'
}
```

For iOS platform

Add source

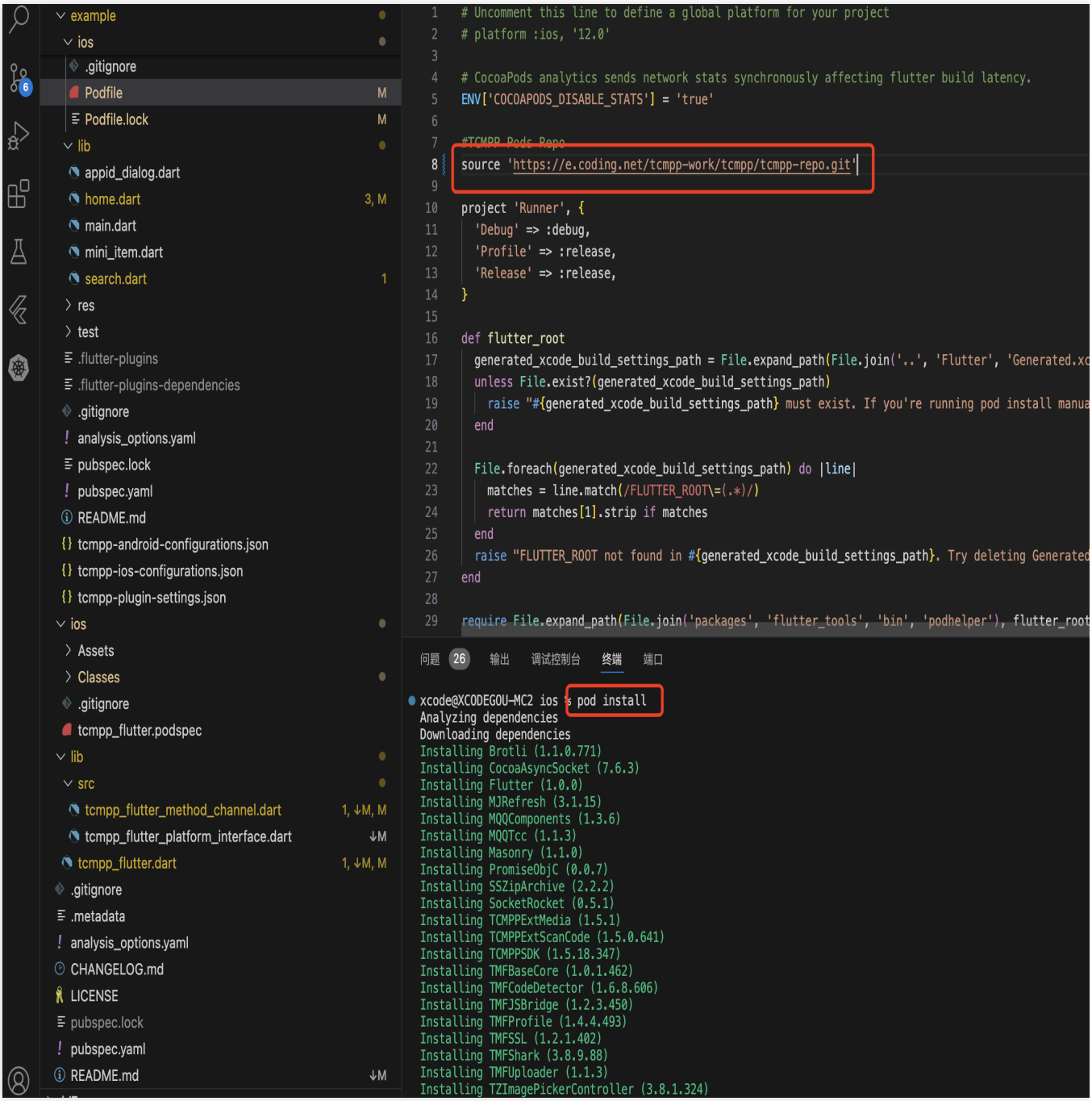
In the Podfile file in the ios directory, add source:

```
source 'https://e.coding.net/tcmpp-work/tcmpp/tcmpp-repo.git'
```

Execute pod install

cd to **ios** directory,execute pod install:

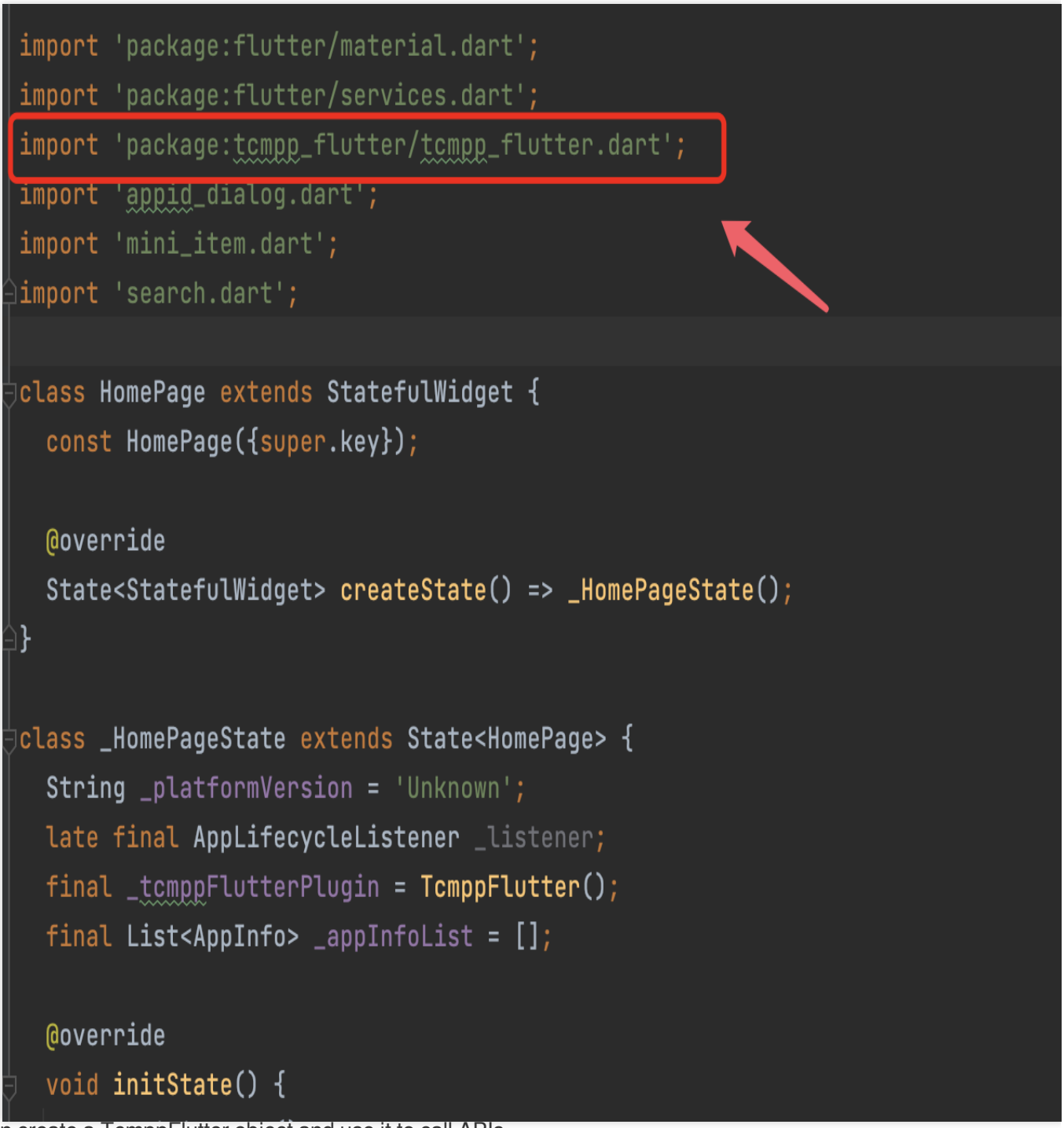
```
pod install
```



Flutter APIs

Usage

1. To use APIs, import 'package:tcmpp_flutter/tcmpp_flutter.dart'.



```dart
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:tcmpp_flutter/tcmpp_flutter.dart';
import 'appid_dialog.dart';
import 'mini_item.dart';
import 'search.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<StatefulWidget> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  String _platformVersion = 'Unknown';
  late final AppLifecycleListener _listener;
  final _tcmppFlutterPlugin = TcmppFlutter();
  final List<AppInfo> _appInfoList = [];

  @override
  void initState() {
```

2. Then create a TcmppFlutter object and use it to call APIs.

```dart
class _HomePageState extends State<HomePage> {
  String _platformVersion = 'Unknown';
  late final AppLifecycleListener _listener;
  final _tcmppFlutterPlugin = TcmppFlutter();
  final List<AppInfo> _appInfoList = [];

  @override
  void initState() {
    super.initState();
    _listener = AppLifecycleListener(onResume: () {
      getRecentMini();
    });
    getRecentMini();
  }

  Future<void> getRecentMini() async {
    List<AppInfo>? infoList =
        await _tcmppFlutterPlugin.getRecentList().catchError((err) {
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: err));
    });
    setState(() {
      _appInfoList.clear();
      if (infoList != null) {
        _appInfoList.addAll(infoList.take(10));
      }
    });
  }
```

Implement Open APIs

Some mini-program APIs need to interact with host app or third-party library, such as get account information or request a payment action. For these APIs to work properly, developer should implement and register a ***OpenApiHandler*** before mini-program is launched. For details about these APIs, see OpenApiHandler in Classes section.

example code:

```
class MyAppletHandler extends OpenApiHandler {
  @override
  Future<Map<String, dynamic>> getUserProfile(
      AppInfo appInfo, Map<Object?, Object?> params) async {
    print("getUserProfile:$appInfo params:$params");
    Map<String, dynamic> result = {
      "userInfo": {
        "nickName": "xcode",
        "avatarUrl":
            "https://staticintl.cloudcachetci.com/cms/backend-cms/8WGP653_%E5%BC%80
        "gender": 1,
        "country": "China",
        "province": "ChongQing",
        "city": "ChongQing",
      }
    };
    return result;
  }

  @override
  Future<Map<String, dynamic>> login(
     AppInfo appInfo, Map<Object?, Object?> params) async {
    /// return current login certification or do login if not logged in
    ...
  }

  @override
  Future<Map<String, dynamic>> checkSession(
     AppInfo appInfo, Map<Object?, Object?> params) async {
     /// throw any error if this api call is failed
     throw NotLoggedInError();
  }

  /// other api implementation
  ...
}
```

The parameter of these API calls are provided by corresponding mini-program API, converted from JSON object into Dart Map object. Also return value of your implementation will be converted into JSON object and delivered to

corresponding mini-program API. You can check out these APIs in the mini program API document.
To register your OpenApiHandler instance, use **registerOpenApiHandler** in TcmppFlutter object.

```
final _tcmppFlutterPlugin = TcmppFlutter();

...

@override
void initState() {
   super.initState();

   ...

   /// Must registered before mini-program is launched
   _tcmppFlutterPlugin.registerOpenApiHandler(MyAppletHandler());
}
```

Implement Custom API

The SDK engine provides an extension mechanism that allows the host app to customize APIs for mini-programs to call.

```
example code:
   /// Mini Program Call
 var opts = {
 api_name: 'myApiName',
 success: function(res) {
   log(res);
 },
 fail: function(res) {
   log(res);
 },
 complete: function(res) {
   log(res);
 },
 data: {
     name : 'kka',
     age : 22
  }
 }
 wx.invokeNativePlugin(opts);
```

```
void initState() {
    super.initState();
    getRecentMini();
    /// Register custom JSAPI names and functions
```

```
    _tcmppFlutterPlugin.registerMiniAppApi("myApiName",myApiHandler);
}


/// Custom API
Future<Map<String, dynamic>?> myApiHandler(MiniApiCall call) async {
    print("API : ${call.apiName}");
    print("AppInfo: ${call.appInfo}");
    print("WebView ID: ${call.webViewId}");
    print("params: ${call.params}");


    return {"result": "success","method":"myApiHandler"};
}
```

Implementing platform event handler

Some platform events will be delivered to Flutter, you can register a handler to subscribe and response to those events.

example code:

```
class MyPlatformHandler extends TcmppPlatformEventHandler {
  @override
  Future<void> onMiniProgramStateChange(
      String appId, MiniProgramState state) async {
    print("app state change: appid=$appId, state=$state");
  }


  /// other event handler implementation
  ...
}
```

```
final _tcmppFlutterPlugin = TcmppFlutter();

...

@override
void initState() {
   super.initState();

   ...

   /// Must registered before mini-program is launched
  _tcmppFlutterPlugin.registerPlatformEventHandler(MyPlatformHandler());
}
```

Implementing a custom menu

Developer can customize buttons in mini-program's action panel. Those include share buttons and other function buttons. Currently, action panel is called out by clicking the more button on top right side of mini-program.
example code:

```
@override
Future<List<CustomMenu>> getCustomMenus() async {
/// The menu includes menuId, picture (supports local path and network picture), me
  CustomMenu menu1 = CustomMenu(
      '100', 'res/images/mini_app_wechat_friend.png', 'Share To', true,
      shareKey: 'twitter');
  CustomMenu menu2 = CustomMenu(
      '101', 'https://staticintl.cloudcachetci.com/cms/backend-cms/8WGP653_%E5%BC%8


  return [
    menu1,
    menu2,
  ];
}
```

```
@override
Future<void> customMenuClick(String menuId, ShareData? shareMenu) async {
  /// callback for menu button click
  print("click menuId:$menuId shareMenu:$shareMenu");
  throw UnimplementedError();
}
```

Mini-program state change event

When mini-program start, go foreground, go background and closed, host Flutter app can be notified by overriding onMiniProgramStateChange method.
example code:

```
@override
Future<void> onMiniProgramStateChange(
    String appId, MiniProgramState state) async {
  print("app state change: appid=$appId, state=$state");
}
```

Log event

If important log message appears, it will also be delivered to host app. Override log method to record these messages.

example code:

```
@override
Future<void> log(String message) async {
  print(message);
}
```

Analysis event

You can capture and record analysis events by implementing reportEvent method. These events including mini-program user interaction activities and crashes.

example code:

```
@override
Future<bool> reportEvent(int eventId, String eventName, AppInfo appInfo,
    Map<Object?, Object?> params) async {
  print("reportEvent:$eventName  appinfo:$appInfo params:$params");
  // TODO: implement reportEvent
  return Future.value(false);
}
```

APIs

```
Future<void> startMiniAppWithId(String appId, MiniStartOptions? options)
```

Start a mini program with given appId

appId: The app id of mini program.

options: Start options of this mini program.

```
Future<void> startMiniAppWithLink(String link, MiniStartOptions? options)
```

Start a mini program with given link

link: The uri link to start.

options: Start options of this mini program.

```
Future<ScanResult?> scan()
```

Start a new page to scan for QR codes, returns a ScanResult if found one.

```
Future<List<AppInfo>?> getRecentList()
```

Get a list of recent opened mini programs

```
Future<List<AppInfo>?> searchMiniApp(String keyword,
   {int pageIndex = 0, int pageSize = 0})
```


Search for online mini programs with given keyword
keyWord: Keyword of mini program to search.
pageIndex: Page to return if pageSize has a none-zero value, which means paging is enabled.
pageSize: The max size of a page returned by server. When set to 0, paging will be disabled.

```
Future<void> stopMiniApp(String appId)
```

Stop mini program with given app ID
appId: The app id of mini program.

```
Future<void> stopAllMiniApp()
```

Stop all running mini programs

```
Future<void> deleteMiniAppCache(String appId,
   {int appVerType = 0, String? ver
```

Remove all local data of mini program with given ID
appId: the app id of mini program
appVerType: type of mini program
version: version of mini program to remove

```
Future<AppInfo?> currentMiniApp()
```

Return information for current foreground running mini program

```
Future<void> preDownloadMiniApp(List<String> appIdList,
   {bool isDownload = false})
```

Preload information for online mini programs
appIdList: a list of app IDs for mini programs to load.
isDownload: if set to true, mini program package will be downloaded after preload, default to false.

```
void registerOpenApiHandler(OpenApiHandler handler)
```

Register handler for sdk to send open api event to your flutter code. These events are triggered when mini-program require third-party information or action, such as, account login, request payment or get user information.
handler: method ref to handle open api event.

```
void registerPlatformEventHandler(TcmppPlatformEventHandler handler)
```

Register handler for sdk to send platform-related event to your flutter code.These events may query information about your flutter application or notify flutter when user interact with mini program container.
handler: method ref to handle platform event

```
void registerMiniAppApi(String apiName, TcmppMiniAppApiHandler apiHandler)
```

Define customised mini program api and associate api handler.apiName: name of your customised api, mini program can use this api by calling wx.(apiNme).

apiHandler: function to handle this api call

```
void setAccount(AccountInfo? info)
```

Set account information of current logged in user for data isolation and user information display

info: information of current logged in account, null if not logged in.

```
Future<void> setLocale(String language,
      {String? region, String? variant})
```

Set localization of mini-programs.must be set before mini-program launched.language: language of mini-programs. Should be an ISO 639 alpha-2 or alpha-3 language code.regin: region code of locale. Should be an ISO 3166 alpha-2 country code.variant: any arbitrary value used to indicate a variation of locale.

```
Future<void> setTheme(MiniTheme theme)
```

Set theme of mini-program container.must be set before mini-program launched.theme: light mode, dark mode or use system setting.

Classes

MiniStartOptions

```
class MiniStartOptions {
  /// entry path of mini program
  String? entryPath;

  /// is always update mini program when start
  bool? isForceUpdate;

  /// string param passed to mini program when start
  String? params;
}
```

ScanResult

```
class ScanResult {
  /// result string of qrcode or barcode contains
  String? result;

  /// the type of code
  String? scanType;
```

```
  /// charset of result string
  String? charset;
}
```

AppInfo

```
class AppInfo {
  /// MiniProgram id
  String appId;

  /// MiniProgram package type (release, dev, etc.). See [AppVerType]
  int appVerType;

  /// MiniProgram version
  String version;

  /// MiniProgram name
  String? name;

  /// MiniProgram icon url
  String? iconUrl;

  /// MiniProgram description
  String? appIntro;

  /// Developer of MiniProgram
  String? appDeveloper;

  /// MiniProgram release time
  int time;
}
```

AppVerType

```
/// Consts for MiniProgram package type. see[AppInfo]
class AppVerType {
  static const int online = 0;
  static const int develop = 1;
  static const int preview = 2;
  static const int experience = 3;
}
```

AccountInfo

```
class AccountInfo {
  /// Unique id of current account
```

```
  String? uid;


  /// Url of avatar image
  String? avatarUrl;


  /// Name of current account
  String? accountName;
}
```

OpenApiHandler

```
abstract class OpenApiHandler {
  /// Called when mini-program invoke wx.requestPayment, requesting third-party pay
  ///
  Future<Map<String, dynamic>> requestPayment(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// Called when mini-program invoke wx.getUserProfile, requesting host app's user
  ///
  Future<Map<String, dynamic>> getUserProfile(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// Called when mini-program invoke wx.login, requesting host app's login certifi
  ///
  Future<Map<String, dynamic>> login(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// Called when mini-program invoke wx.checkSession, requesting host app's login
  /// check if login expired
  ///
  Future<Map<String, dynamic>> checkSession(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// Called when mini-program invoke wx.getUserInfo, deprecated by wx.getUserProfi
  /// compatibility for earlier mini-program api
  ///
  Future<Map<String, dynamic>> getUserInfo(
      AppInfo appInfo, Map<Object?, Object?> params);

  /// Called when mini-program invoke wx.getPhoneNumber, get current user's phone n
  ///
  Future<Map<String, dynamic>> getPhoneNumber(
      AppInfo appInfo, Map<Object?, Object?> params);
}
```

TcmppPlatformEventHandler

```
abstract class TcmppPlatformEventHandler {


  Future<String> getAppName() async {
    return "";
  }


  Future<String> getAppVersion() async {
    return "";
  }


  Future<void> log(String message) async {}


  Future<List<CustomMenu>> getCustomMenus() async {
    return [];
  }


  Future<void> customMenuClick(String menuId, ShareData? shareMenu) async {}

  Future<void> onMiniProgramStateChange(
      String appId, MiniProgramState state) async {}


  Future<bool> reportEvent(int eventId, String eventName, AppInfo appInfo,
      Map<Object?, Object?> params) async {
    return false;
  }
}
```

App Server
Header Information

Last updated：2025-04-03 11:36:37

Note:

Supported only on public cloud.

Common request header information

Request header	Description	Note	Example
Content-Type	application/json	-	-
TC-Timestamp	Timestamp in milliseconds.	This field is provided by Super App as a Service (SAS).	1735293931220
TC-Signature	Signature string.	This field is provided by Super App as a Service (SAS) and is a hex-encoded string based on AES CBC encryption of the TC-Timestamp value string. The secret key is referenced in the Configuration Management .	xxx

Signature algorithm

Advanced Encryption Standard (AES)

Example

```
current := timeutils.GetCurrentTimestamp()
Signature:=cryptoutils.InnerEncrypt(current,secret)
```

```
func InnerEncrypt(data, secretKey string) (string, error) {
    aesKey := []byte(secretKey)
    block, err := aes.NewCipher(aesKey)
    if err != nil {
       return "", err
    }
    blockSize := block.BlockSize()
    origData := c.PKCS7Padding([]byte(data), blockSize)
    blockMode := cipher.NewCBCEncrypter(block, aesKey[:blockSize])
    encipheredData := make([]byte, len(origData))
    blockMode.CryptBlocks(encipheredData, origData)
    return hex.EncodeToString(encipheredData), nil
}
```

Payment Development Notes
Mini Program Payment: Parameter Application and Configuration
Required Development Parameters

Last updated：2025-04-03 11:36:38

Before starting development in the standard merchant mode, the following development required parameters need to be prepared:

Parameter	Description
mchid	Merchant ID (mchid) is the unique identifier for the merchant in the superapp's payment. All payment-related API calls must include this parameter for the superapp's payment to verify the merchant's identity. This ID is provided to the merchant upon successful registration. For details, see Mini Program Payment.
serial_no	Merchant certificate serial number.
appid	A unique identifier applied for on Super App as a Service (SAS). To use the superapp's payment, this ID must be bound to the merchant ID. Refer to the platform's merchant ID binding process for details.
APIv3 key	The superapp's payment uses the APIv3 key to encrypt callback information. Merchants must decrypt the encrypted callback data using this key. Set up the APIv3 key before integration.
Merchant API certificate	Used to generate request signatures with the private key when initiating APIv3 requests.
Superapp's payment public key	Used to verify signatures on APIv3 responses and encrypt sensitive information (e.g., names, ID numbers) for transmission.

The superapp needs to enable the following features:

1. Enable merchants to complete the onboarding process and provide them with mchid (merchant ID), serial_no (merchant certificate serial number), and merchant API certificate.
2. Provide a feature for merchants to set up the APIV3 key.
3. Allow merchants to request and get the superapp's payment public key.

Configure API Key

Last updated：2025-04-03 11:36:38

The APIv3 key is mainly used for decrypting platform certificates and callback information.
The superapp should enable merchants to configure their own APIv3 key, which will be used to encrypt callback request messages after successful payments.

Parameter Application

Last updated：2025-04-03 11:36:38

1. Apply for a merchant ID

Each merchant ID is tied to one settlement currency. To receive payments in multiple currencies, you must apply for additional merchant IDs.
You can apply for a merchant ID and a mini program ID simultaneously.
Once the application is successful, a notification email containing the merchant ID and its login credentials will be sent to the contact email provided by the merchant. Please keep this information secure.

2. Apply for a mini program ID

Get the mini program ID by creating a mini program on Super App as a Service (SAS).

3. Bind the mini program ID and the merchant ID

After applying for both the mini program ID and merchant ID, you need to establish a binding relationship between them. (Refer to Super App as a Service (SAS) merchant binding process for details.)
In the standard merchant mode, the relationship between mini program ID and merchant ID is many-to-many, meaning one mini program ID can bind multiple merchant IDs, and one merchant ID can bind multiple mini program IDs.

Mini Game Payment: Parameter Application and Configuration
Parameter Application

Last updated：2025-04-22 18:11:24

1. Apply for a merchant ID

Each merchant ID is tied to one settlement currency. To receive payments in multiple currencies, you must apply for additional merchant IDs.
You can apply for a merchant ID and a mini game ID simultaneously.
Once the application is successful, a notification email containing the merchant ID and its login credentials will be sent to the contact email provided by the merchant. Please keep this information secure.

2. Apply for a mini game ID

Get the mini game ID by creating a mini game on Super App as a Service (SAS).

3. Bind the mini game ID and the merchant ID

After applying for both the mini game ID and merchant ID, you need to establish a binding relationship between them. (For details, refer to Bind a merchant account)
In the standard merchant mode, the relationship between mini game ID and merchant ID is many-to-many, meaning one mini game ID can bind multiple merchant IDs, and one merchant ID can bind multiple mini game IDs.

Development Required Parameters

Last updated：2025-04-22 18:11:24

Before starting development in the standard merchant mode, the following development required parameters need to be prepared:

Parameter	Description
mchid	Merchant ID (mchid) is the unique identifier for the merchant in the superapp's payment. All payment-related API calls must include this parameter for the superapp's payment to verify the merchant's identity. This ID is provided to the merchant upon successful registration. For details, see Virtual payment-Direct Purchase of Mini Game Items.
AppKey	It is used to sign the payment request, indicating that the request is initiated by the payment module on the developer's server. For details, refer to Payment request signature algorithm

The superapp needs to enable the following features:

1. Enable merchants to complete the onboarding process and provide the mchid (merchant ID) to the merchant.
2. Provide a feature for merchants to set up the AppKey.

Mini Program Payment: Signature and Verification
Superapp Payment Certificate Signature

Last updated：2025-04-03 11:36:38

The superapp's payment uses its certificate's private key to perform SHA256 with RSA signing on the "response signature string," and then encodes the signature result using Base64 to obtain the signature value..

Construct the signature string

1. Get the following information from the response or notification callback:
HTTP Header Wechatpay-Timestamp: The response timestamp.
HTTP Header Wechatpay-Nonce: The response nonce.
Response body: Use the original response body for verification. Ensure that any framework used does not alter the response body, as any modification will result in verification failure.
2. Construct the response signature string according to the following rules. The signature string consists of three lines, each ending with a newline character (\\n, ASCII value 0x0A). If the response body is empty (e.g., HTTP status code 204 No Content), the last line will be just a newline character.

```
Request timestamp\\n
Request random string\\n
Request body\\n
```

Get the response signature

The response signature from the supperapp's payment is transmitted via the HTTP header Wechatpay-Signature. (Note: Examples may contain line breaks, but actual data should be in one line). Use Base64 decoding on the Wechatpay-Signature field value to obtain the response signature.

Verify the signature

Use the superapp's payment public key to verify the signature string and SHA256 with RSA signature.

Decrypt Callback Messages

Last updated：2025-04-03 11:36:38

To ensure security, the superapp's payment encrypts critical information using AES-256-GCM in the callback notification API. Upon receiving the message, merchants need to decrypt it to obtain the plaintext. The APIv3 key is the symmetric key used for decryption. This document describes the format of the encrypted message and the decryption process.

1. Encrypted message format

AES-GCM is a NIST standard authenticated encryption algorithm that ensures data confidentiality, integrity, and authenticity. It is most widely used in TLS connections.
The encryption key used for certificates and callback messages is the APIv3 key. For details, see Development Parameter Application and Configuration.
For the encrypted data, we use a separate JSON object to represent it. For ease of reading, the example is pretty-formatted and includes comments.

```
{
  "original_type": "transaction", // The type of object before encryption
  "algorithm": "AEAD_AES_256_GCM", // Encryption algorithm
  // Base64-encoded ciphertext
  "ciphertext": "...",
  // Random string initialization vector used for encryption
  "nonce": "...",
  // Additional data packet (may be empty)
  "associated_data": ""
}
```

2. Decryption

For details of the algorithm API, see RFC 5116.
Most programming languages (newer versions) support AEAD_AES_256_GCM. You can refer to the following examples to understand how to implement decryption in your programming language.
JAVA：

```
import java.io.IOException;
import java.security.GeneralSecurityException;
import java.security.InvalidAlgorithmParameterException;
```

```
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.GCMParameterSpec;
import javax.crypto.spec.SecretKeySpec;
public class AesUtil {
  static final int KEY_LENGTH_BYTE = 32;
  static final int TAG_LENGTH_BIT = 128;
  private final byte[] aesKey;
  public AesUtil(byte[] key) {
    if (key.length != KEY_LENGTH_BYTE) {
      throw new IllegalArgumentException("Invalid ApiV3Key. The length must be 32 b
    }
      this.aesKey = key;
  }
  public String decryptToString(byte[] associatedData, byte[] nonce, String ciphert
  throws GeneralSecurityException, IOException {
    try {
      Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
      SecretKeySpec key = new SecretKeySpec(aesKey, "AES");
      GCMParameterSpec spec = new GCMParameterSpec(TAG_LENGTH_BIT, nonce);
      cipher.init(Cipher.DECRYPT_MODE, key, spec);
      cipher.updateAAD(associatedData);
      return new String(cipher.doFinal(Base64.getDecoder().decode(ciphertext)), "ut
    } catch (NoSuchAlgorithmException | NoSuchPaddingException e) {
      throw new IllegalStateException(e);
    } catch (InvalidKeyException | InvalidAlgorithmParameterException e) {
      throw new IllegalArgumentException(e);
    }
  }
}
```

Python:

```
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
import base64
def decrypt(nonce, ciphertext, associated_data):
    key = "Your32Apiv3Key"
    key_bytes = str.encode(key)
    nonce_bytes = str.encode(nonce)
    ad_bytes = str.encode(associated_data)
    data = base64.b64decode(ciphertext)
    aesgcm = AESGCM(key_bytes)
    return aesgcm.decrypt(nonce_bytes, data, ad_bytes)
```

GO:

```
// DecryptGCM decrypts ciphertext using AES-256-GCM
func DecryptGCM(ciphertext []byte, key []byte, nonce []byte, additionalData []byte)
   block, err := aes.NewCipher(key)
   if err != nil {
      return nil, err
   }
   gcm, err := cipher.NewGCM(block)
   if err != nil {
      return nil, err
   }


   plaintext, err := gcm.Open(nil, nonce, ciphertext, additionalData)
   if err != nil {
      return nil, err
   }
   return plaintext, nil
}
```

Initiate Payment Signature

Last updated：2025-04-03 11:36:38

When initiating payments, merchants need to use the merchant API certificate to create the payment signature. We'll guide you step-by-step using test public and private keys to generate a "mini program payment" signature. We recommend testing with these keys to ensure your signature matches the document. For real transactions, use your actual parameters.

Get the merchant API certificate

The merchant API certificate is required to construct a request signature. For details, see Required Development Parameters.
We have got the merchant API certificate and stored the corresponding merchant API private key (apiclient_key.pem) locally.

Merchant ID: 202003191046
Certificate serial number: 526807E51G82219FOC2D5D3E6AB8ED1S8SDS8787AS
API certificate private key: Save the following file in PEM format. To avoid confusion with your actual public and private keys, we have saved it as apiclient_test_key.pem.
The following is a sample key and is not actually usable.
-----BEGIN PRIVATE KEY-----
\\nMIIEuwIBADALBgkqhkiG9w0BAQEEggSnMIIEowIBAAKCAQEA1WFrv7DQ2FeBB2ZR\\n/bh+W/38+Rgcs/yxTdd0/9r5DWYvB6Lhc0pqaNnrmZUc+Uih6CELe1K3AAvg0+6A\\njfcLV/aqNx4xdwqfLt2P1F7TsyGJZWMe5OoPmwzel8zRpGcqY/WdmevEEFqmIc/r\\nWVa1fOCM9eIzP9QQbgT7tKXa/ixi5B5y8B0pShYJuyE2M8GimvbDbnaatMQJlwFP\\nO9fxU7cRZBdkDcUB8dMxl2ZfTHZjEC6ypR4Ux5vnPIB9hH7qHFbc6W9ueEfVRTk3\\neeoVnbmZJHliWBqtv89Tm0uMk+fD2ZayRA+TuwFajt0NTcbnM6kM0cuuyEyd7bnE\\nWFKuAwIDAQABAoIBACS+8CVEt9Jpz0iM8FW3Ldt9s9DZvCeqvoXfMsDU3srV7Adu\\nn1CRYh3IWXBLY3/yaB9ngWitZ+JUKVWV3wGTp5pwWgO/6VjMtXkGorw50E8q2VRi\\na3GUdTeIUdTmarvbIEuygn99QHhog++StL7f1cU5jkzRtW2qgWHQ7d/AKCRZA+R1\\nnUwNaQHdz2Fn5a5cQsULgNCf0Rfn4MxgsvGl3ZVcJVUiumEDfV2TDcLz2wEaWvTo\\nOhD6bN+Ug0LuucmuwC9FzR7DUNxWxmQpAdPMbAfku47K9ARqHfUjNXtBUktGdo6x\\nfmdm/fNTodSzziu4Sn87iQU0R7VU8TT2Wx1l/jECgYEA3/3yqEWSwjCY/hs+rq9O\\nrhF4vVyd8az7X+KCKiYZl51oRiRSso8dWvuVixpx3ZW8vp81K2eq9h6BmuePMVZK\\nH8PV8LbNbuLUn/cTREo7JcT0jUFSfwyMiu6De23fyCSb3fM4EFdjuywTn0d+RIr5\\nlnurFc8mRWTTX0E+kht7K50CgYEA899JYDMqs7GU+Gg8vNEHL3ux4VIGWaV0LPFj\\nn4UNn0aT3t0M+OgWm9K1tCSi5PPkmkAt8wCOtKPmSiq1CQeWa8HX+JHkMiEYO6Ki\\nHecXmZlUr/yXMhCTkkxwNsFAFxP1KYOm91+ka6w+l7/qcjan+WZsYT2XpSTx0LV5\\nPma8Hh8CgYABUNuZE3eOPnzXmU9f9VWv/hhIfH/NCKgdYxZCqyChXGJdbx8xP1f7\\nzdiODaS3mYaXVBYa4CwH8BvwzgVwU8Jxt1PNazV/vkNjgS8SyqDYUvTg045pgqhc\\ntJP/KKEU6uojfqdIqUrDsbmXyPK78lkPAkD6CtJ9u97mA1sbvp+VnQKBg

Fp41qba\\ntJfPZJ23RfkibtD9yaL2pCZzzCK0NqpCWShirY77YMmiiGishf5brRbVKFTVRHan\\nGUoIl/Gh4GGGMBav5ihwL0Etp+jPz+baCZZRHOrhAVJwdd7LfsHBdb5aCBSro7CY\\nCc5sKxhu+VH/1tceWUzF5dE9YHx2JpGw2U8vAoGBAL2Wp4S2dA+zKfhX7QOCLl3q\\nXYujhL1dgZBaDonWtOrn7llLSqaryD/TH8C6QRVrsXpLdwuSLx7tzQnG81ptO49Q\\nuCVFbGF5RwCf8Wq8OlYuJ/MS9GsE+Ux2EYVX3DD5zV6gtN11c7NsTEan9fRpgZjt\\n2kuvKl1oec/Rh8fbmqid\\n-----END PRIVATE KEY-----

Construct the signature string

Below is the specific format of the signature string.

The signature string consists of four lines, with each line representing a parameter. It ends with a newline character (\\n, ASCII value 0x0A), including the last line.

```
Mini program ID\\n
Timestamp\\n
Random string\\n
prepay_id\\n
```

Take calling the payment initiation API in the mini program as an example.

Step 1: Get the mini program ID. For the definition of a mini program ID and how to get it, see Required Development Parameters.

mpco56h12e6e52hj

Step 2: Get the current system timestamp when initiating the request. This is the total number of seconds from 00:00:00 GMT on January 1, 1970 (08:00:00 Beijing time on January 1, 1970) to the present. The superapp's payment will reject requests made a long time ago, so merchants should ensure their system time is accurate.

1$ date +%s

1742351329

Step 3: Generate a request random string.

E6F165123B4E32D8D0D6

Step 4: Get the prepay_id.

Calling the order placement API will return the prepay_id.

prepay_id=pip17423513210901nlprl31haz5fdnxqoty

Step 5: Construct the request signature string according to the rules above.

mpco56h12e6e52hj\\n

1742351329\\n

E6F165123B4E32D8D0D6\\n

pip17423513210901nlprl31haz5fdnxqoty\\n

Calculate the signature value

Most programming languages provide functions to sign data. We strongly recommend that merchants use these functions to perform SHA256 with RSA signing on the signature string using their private key, and then Base64

encode the result to get the signature value.

Below is the resulting signature value. Use verification tools to ensure your signature matches this one. If they match, the calculation is correct. If not, check the parameters and ensure there are no unintended line breaks.

uVOT92EjeLfCjQz8gXYfjgPaSVxOoAcwzalWcahkzCPw3QpAKrVmBbTa/d6hx/JfPm1ajBnLmasDRcu93f8nn2LoOMvDSkSu4T7WJW0CD2XRBdr7GN2WrIftd17iCBqOjWKgo4dIQtnx/gGThqw0UZ2zsHqtItk6N1VN+9OksDJSJF806KxBAjG+yljvjFQZ1nV6ergVWDqPJ9KghdPhILCzwQy9Vbc3jQJFKFJPJfiHdVAzoikv0rlLV/22CF4dxs0iSqLRrZuu9IbEc9JOMGXbyrxmHcwgXfe/+Rc71KHYDvqIWxhAkxvLUkJrtzWx4bwAfNyMzniN1XHthpzN3g==

Example

```
wx.requestPayment
(
  {
    "timeStamp": "1742351329",
    "nonceStr": "E6F165123B4E32D8D0D6",
    "package": "prepay_id=pip17423513210901nlprl31haz5fdnxqoty",
    "signType": "RSA",
    "paySign": "uVOT92EjeLfCjQz8gXYfjgPaSVxOoAcwzalWcahkzCPw3QpAKrVmBbTa/d6hx/JfPm1
    "success":function(res){},
    "fail":function(res){},
    "complete":function(res){}
  }
)
```

APIv3 Certificate Signature

Last updated：2025-06-27 17:09:11

Overview

Developers must sign and verify signatures for all request-response scenarios, API callbacks, and payment initiation within the superapp's payment API.

When is a signature required? How to sign?

Request the superapp's payment API

Merchants need to use their API certificate to create request signatures when calling the superapp's payment API. The signing method varies based on the request parameters. Refer to the following guidelines for details.

How to create a request signature

Merchant API certificate

The merchant API certificate is required to create a request signature.
We have got the merchant API certificate and stored the corresponding merchant API private key (apiclient_key.pem) locally.

Merchant ID: 202003191046
Certificate serial number: 526807E51G82219FOC2D5D3E6AB8ED1S8SDS8787AS
API certificate private key: Save the following file in PEM format. To avoid confusion with your actual public and private keys, we have saved it as apiclient_test_key.pem.
Note:
The following is a sample key and is not actually usable.

```
-----BEGIN PRIVATE KEY-----\\nMIIEuwIBADALBgkqhkiG9w0BAQEEggSnMIIEowIBAAKCAQEA1WFrv
```

Construct the signature string

We ask that the merchant's technical developers construct the signature string according to the rules outlined in this document. The superapp's payment will use the same method to construct the signature string. If the merchant constructs the signature string incorrectly, the signature verification will fail. Below is the specific format of the signature string.

The signature string consists of five lines, with each line representing a parameter. It ends with a newline character (\\n, ASCII value 0x0A), including the last line. If a parameter itself ends with \\n, an additional \\n should be appended.

```
HTTP request method\\n
URL\\n
Request timestamp\\n
Request random string\\n
Request body\\n
```

1. Request parameters with path parameters

Take the order query API as an example.

Step 1: Get the HTTP request method.

GET

Step 2: Get the absolute URL of the request.

The URL for the query order API is /v3/pay/transactions/out-trade-no/{out_trade_no}. The path includes the parameter out_trade_no, which should be replaced with the actual merchant order number, e.g., 1217752501201407033233368018.
/v3/pay/transactions/out-trade-no/1217752501201407033233368018

Step 3: Get the current system timestamp when the request is initiated.

This is the total number of seconds from 00:00:00 GMT on January 1, 1970 (08:00:00 Beijing time on January 1, 1970) to the present. The superapp's payment will reject requests made a long time ago, so merchants should ensure their system time is accurate.
date +%s
1554208460

Step 4: Generate a request random string.

E6F165123B4E32D8D0D6

Step 5: Get the request body.

For this API, the request body is an empty string, so just append a \\n.

Step 6: Construct the request signature string according to the rules above.

```
POST\\n
/v3/pay/transactions/jsapi\\n
1554208460\\n
E6F165123B4E32D8D0D6\\n
{"appid":"wxd678efh567hg6787","mchid":"1230000109","description":"Image","out_trade
```

2. Request parameters with body parameters

Take the order API as an example.

Step 1: Get the HTTP request method.

POST

Step 2: Get the absolute URL of the request, excluding the domain part.

/v3/pay/transactions/jsapi

Step 3: Get the current system timestamp when the request is initiated.

This is the total number of seconds from 00:00:00 GMT on January 1, 1970 (08:00:00 Beijing time on January 1, 1970) to the present. The superapp's payment will reject requests made a long time ago, so merchants should ensure their system time is accurate.
date +%s
1554208460

Step 4: Generate a request random string.

E6F165123B4E32D8D0D6

Step 5: Get the request body.

You can place all parameters on one line; the body parameter in the signature request should also be on one line. Alternatively, you can calculate the signature with parameters on multiple lines; the body parameter in the request should match this format. Essentially, the body in the signature calculation should be identical to the body in the request.
Here is an example with all parameters on one line:

```
{"appid":"wxd678efh567hg6787","mchid":"1230000109","description":"describe","out_tr
```

Step 6: Construct the request signature string according to the rules above.

```
POST\\n
/v3/pay/transactions/jsapi\\n
1554208460\\n
E6F165123B4E32D8D0D6\\n
{"appid":"wxd678efh567hg6787","mchid":"1230000109","description":"describe","out_tr
```

When the request body is an empty string, simply append a \\n.

3. Request parameters query parameters

Take querying the order by merchant order number as an example:

Step 1: Get the HTTP request method.

GET

Step 2: Get the absolute URL of the request, excluding the domain part.

Concatenate your query parameters. Suppose your query parameters are as follows:

```
limit=52offset=103authorized_data={"business_type":"FAVOR_STOCK", "stock_id":"24334
```

(1) First, URL encode the authorized_data and partner parameters:

```
limit=52offset=103authorized_data%3D%7B%22business_type%22%3A%22FAVOR_STOCK%22%2C%2
```

(2) Concatenate your request URL. The query parameters need to be appended at the end with '?' and the corresponding query string. Multiple strings should be linked with & (Note: The following URL is on one line; due to formatting, it may appear as multiple lines, but the actual data should be on one line).
/v3/pay/transactions/out-trade-no/112233445566?
limit=5&offset=10&authorized_data%3D%7B%22business_type%22%3A%22FAVOR_STOCK%22%2C%20%22stock_id%22%3A%222433405%22%7D&partner%3D%7B%22type%22%3A%22APPID%22%2C%22appid%22%3A%22wx4e1916a585d1f4e9%22%2C%22merchant_id%22%3A%222480029552%22%7D

Step 3: Get the current system timestamp when the request is initiated.

This is the total number of seconds from 00:00:00 GMT on January 1, 1970 (08:00:00 Beijing time on January 1, 1970) to the present. The superapp's payment will reject requests made a long time ago, so merchants should ensure their system time is accurate.
date +%s
1554208460

Step 4: Generate a request random string.

E6F165123B4E32D8D0D6

Step 5: Get the request body.

The request body is an empty string, simply append a \\n.

Step 6: Construct the request signature string according to the rules above.

When the request body is an empty string, simply append a \\n.

GET\\n /v3/marketing/partnerships?
limit=5&offset=10&authorized_data%3D%7B%22business_type%22%3A%22FAVOR_STOCK%22%2C%20%22stock_id%22%3A%222433405%22%7D&partner%3D%7B%22type%22%3A%22APPID%22%2C%22appid%22%3A%22wx4e1916a585d1f4e9%22%2C%22merchant_id%22%3A%222480029552%22%7D\\n 31554208460\\n
E6F165123B4E32D8D0D6\\n \\n

Calculate the signature value

Most programming languages provide functions to sign data. We strongly recommend that merchants use these functions to perform SHA256 with RSA signing on the signature string using their private key, and then Base64 encode the result to get the signature value.

Please pay attention to handling single and double quote escape issues. If the outer layer of the second line uses single quotes, the inner parameters do not need to be escaped. If the outer layer uses double quotes, then the double quotes within the body parameter need to be escaped.

1. Set the HTTP Authorization Header

The request passes the signature through the HTTP Authorization Header. The Authorization Header consists of two parts: the authentication type and the signature information.

Below, we demonstrate how to generate the signature using the command line.

Authorization Header: Authentication type and signature information

The specific composition is:

2. Authentication type. currently WECHATPAY2-SHA256-RSA2048

3. Signature information. For parameters, see Required Development Parameters

Merchant ID (mchid) of the merchant initiating the request (including direct merchants, service providers, or channel merchants).

Merchant API certificate serial number (serial_no), used to declare the certificate being used.

Request nonce (nonce_str), which should be consistent with the nonce string used in the signature string construction.

Timestamp (timestamp), which should be consistent with the timestamp used in the signature string construction.

Signature value (signature), calculated as described above.

Note：

The above five signature information items do not have a specific order requirement.

An example of the Authorization Header is as follows. (Note that the example may have line breaks due to formatting, but the actual data should be in one line):

```
Authorization Header: WECHATPAY2-SHA256-RSA2048 mchid="1900007291",nonce_str="593BE
```

Finally, we can construct an HTTP request that includes the signature.

(1) Please note that the body parameter in the sixth line must be on one line and cannot have line breaks, as the signature calculation was done with the body on one line. The request must be consistent with the signature calculation.

(2) Please ensure that the timestamp="1554208460" and serial_no="408B07E79B8269FEC3D5D3E6AB8ED163A6A380DB" in the Authorization Header are consistent with the values used in the signature calculation.

(3) This is just an example to illustrate the format for reference. Since the example key itself is not functional, the following request is not actually usable.

```
curl -X POST \\
  https://api.mch.weixin.qq.com/v3/pay/transactions/jsapi \\
  -H 'Authorization: WECHATPAY2-SHA256-RSA2048 mchid="202003191046",nonce_str="E6F1
```

```
    -H 'Accept: application/json' \\
    -H 'Content-Type: application/json' \\
    -d '{"appid":"mp1bfa1hnwvaluqb","mchid":"202003191046","description":"goods desc"
```

Mini Game Payment Signature and Verification

Last updated：2025-04-22 18:11:24

Payment request signature algorithm

The pay_sig parameter uses a signature algorithm that signs the payment request with the AppKey obtained from the superapp. This indicates that the request is initiated by the developer's server-side payment module. The signature algorithm pseudocode is as follows:

```
paySig = to_hex(hmac_sha256(appKey,'requestMidasPaymentGameItem' + '&' + signData))
```

```
  import hmac
  import hashlib
  import urllib.parse

  # sign_data: The original payment string. Note that sign_data must be exactly the
  # appkey: The secret key
  # method: The signature method, which should be requestMidasPaymentGameItem
  def gen_pay_sig(sign_data, appkey, method):
      need_encode_body = method + '&' + sign_data
      print(need_encode_body)
      return hmac.new(key=appkey.encode('utf-8'), msg=need_encode_body.encode('utf-
              digestmod=hashlib.sha256).hexdigest()
```

User session signature

Some server-side APIs require session_key for user authentication. To keep the session_key confidential, it is not transmitted in plaintext. Instead, the API uses a session signature for verification.
The signature is generated as follows: signature = hmac_sha256(session_key, rawData). Here, rawData refers to the parameters sent by the developer when calling the server-side API.

Payment subscription event signature algorithm

The pay_event_sig parameter uses a signature algorithm that signs the payment event request with the AppKey from the superapp, indicating the request is initiated by the developer's server-side payment module. The signature algorithm pseudocode is as follows:

```
pay_event_sig = to_hex(hmac_sha256(app_key, event + '&' + payload))
```

You can refer to the following Python example for the implementation of calc_pay_event_sig:

- event is the type of event being pushed, e.g., minigame_coin_deliver_completed.

- app_key is the AppKey configured in the superapp.
- payload is the data being pushed, corresponding to the payload in the mini game structure. Refer to the specific push request parameter description.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
""" Example for calculating PayEventSig signature """

import hmac
import hashlib


def calc_pay_event_sig(event, payload, appkey):
    """ pay_event_sig signature algorithm
      Args:
          event - Event type, e.g., minigame_game_pay_goods_deliver_notify
          payload - Event payload, the payload in the notification message, e.g.,{"
          app_key - AppKey configured in the superapp
      Returns:
          Payment request signature pay_event_sig
    """
    need_sign_msg = event + '&' + payload
    pay_sig = hmac.new(key=appkey.encode('utf-8'), msg=need_sign_msg.encode('utf-8'
                       digestmod=hashlib.sha256).hexdigest()
    return pay_sig
```

API List

Last updated：2025-06-27 18:00:53

Note:

Supported only on public cloud.

1. Mini program login APIs

1.1 Check whether the user exists

Path: /superapp/inner/user/checkUser

Method: POST

API description:

Checks if the user exists based on the superapp's user ID.

Request parameters

Body

Name	Type	Required	Description
userId	string	True	The anonymized user ID, which is generated by the superapp using the SDK and is used to temporarily request the generation of an anonymized openid from the superapp's server. It is temporarily used when the mini program needs to obtain the openid.

Response parameters

Name	Type	Required	Description
returnCode	string	True	Response code, 0 indicates success.
returnMessage	string	False	Response information.
data	Boolean	True	Indicates whether the user exists.
requestId	string	True	Request trace ID.

1.2 Get user temporary information code

Path: /superapp/inner/user/getUserInfoTemporaryCode

Method: GET

API description:

Gets a temporary credential for the user's mobile number or email based on the type.

Request parameters

Body

Name	Type	Required	Description
type	string	True	The type of information. Valid values: email, phone.
userId	string	True	The anonymized user ID, which is generated by the superapp using the SDK and is used to temporarily request the generation of an anonymized openid from the superapp's server. It is temporarily used when the mini program needs to obtain the openid.

Response parameters

Name	Type	Required	Description
returnCode	string	True	Response code, 0 indicates success.
returnMessage	string	False	Response information.
data	object	True	Response data.
data.data	string	True	Returns the masked mobile number or email based on the query type, e.g., 158****2850, mu****ng@tencent.com.
data.code	string	True	Gets the temporary credential code for the mobile number or email.
RequestId	string	True	Request trace ID.

1.3 Get user email

Path: /superapp/inner/user/getUserEmail

Method: GET

API description:

Gets the user's email based on the temporary credential code.

Request parameters

Body

Name	Type	Required	Description

temporaryCode	string	True	Temporary credential code.
userId	string	True	The anonymized user ID, which is generated by the superapp using the SDK and is used to temporarily request the generation of an anonymized openid from the superapp's server. It is temporarily used when the mini program needs to obtain the openid.

Response parameters

Name	Type	Required	Description
returnCode	string	True	Response code, 0 indicates success.
returnMessage	string	False	Response information.
data	string	True	User email information, a Base64 string after being encrypted using AES CBC. The secret key is referenced in the Configuration management .
requestId	string	True	Request trace ID.

1.4 Get user’s mobile number

Path: /superapp/inner/user/getUserPhoneNumber

Method: GET

API description:

Gets the user's mobile number based on the temporary credential code.

Request parameters

Body

Name	Type	Required	Description
temporaryCode	string	True	Temporary credential code.
userId	string	True	The anonymized user ID, which is generated by the superapp using the SDK and is used to temporarily request the generation of an anonymized openid from the superapp's server. It is temporarily used when the mini program needs to obtain the openid.

Response parameters

Name	Type	Required	Description

returnCode	string	True	Response code, 0 indicates success.
returnMessage	string	False	Response information.
data	string	True	User mobile number, a Base64 string after being encrypted using AES CBC. The secret key is referenced in the Application Management .
requestId	string	True	Request trace ID.

1.5 Get user’s nickname

Path: /superapp/inner/user/getUserNick

Method: GET

API description:

Gets the user's nickname.

Request parameters

Body

Name	Type	Required	Description
userId	string	True	The anonymized user ID, which is generated by the superapp using the SDK and is used to temporarily request the generation of an anonymized openid from the superapp's server. It is temporarily used when the mini program needs to obtain the openid.

Response parameters

Name	Type	Required	Description
returnCode	string	True	Response code, 0 indicates success.
returnMessage	string	False	Response information.
data	string	True	User’s nickname.
requestId	string	True	Request trace ID.

1.6 Get user’s profile photo

Path: /superapp/inner/user/getUserAvatar

Method: GET

API description:

Gets the user's profile photo.

Request parameters

Body

Name	Type	Required	Description
userId	string	True	The anonymized user ID, which is generated by the superapp using the SDK and is used to temporarily request the generation of an anonymized openid from the superapp's server. It is temporarily used when the mini program needs to obtain the openid.

Response parameters

Name	Type	Required	Description
returnCode	string	True	Response code, 0 indicates success.
returnMessage	string	False	Response information.
data	string	True	URL of the user’s profile photo.
requestId	string	True	Request trace ID.

Error codes

Error code	Error code value	Description
0	OK	-
200000	request data is not exist	The request data does not exist.
200001	Invalid User	User error or exception.
200002	Invalid User id	Invalid user ID.
200003	ITemporary code error or expired	Temporary code error or expired.

1.7 Receive subscription messages

Path: /superapp/message/send

Method: POST

API description:

Receives user-subscribed messages pushed from Super App as a Service (SAS).

Request parameters

Body

Name	Type	Required	Description
AccountId	string	True	The owner user ID of the message (same as UserId).
MessageId	string	True	Unique ID of the message.
Content	string	True	Message content.
DataTime	int	True	Timestamp of the message sending time (in seconds).
TemplateId	string	True	Message template ID.
MnpId	string	True	Mini program ID.
MnpName	string	True	Mini program name.
TemplateTitle	string	True	Template title.
State	string	True	The type of mini program to navigate to: developer for the development version, trial for the Preview, formal for the official version. Defaults to the official version.
Page	string	True	The page to navigate to when the template card is clicked. This must be a page within the mini program. Parameters can be included (e.g. index?foo=bar). If not provided, the template will not have a navigation link.
MnpIcon	string	True	Mini program icon.

Response parameters

Name	Type	Required	Description
ReturnCode	string	True	Response code, 0 indicates success.
ReturnMessage	string	False	Response information.
Data	bool	True	Processing result.
RequestId	string	True	Request trace ID.

Error codes

Error code	Error code value	Description
0	OK	-
200000	request data is not exist	The request data does not exist.
200001	Invalid User	User error or exception.
200002	Invalid User id	Invalid user ID.
200003	ITemporary code error or expired	Temporary code error or expired.

2. Mini program payment APIs

2.1 Create a mini program order

Path: /v3/pay/transactions/jsapi

Method: POST

API description:

Creates a mini program order.

Request parameters

Headers

Parameter	Description	Required	Description
TC-Payment-Callback	The callback URL of a successful or failed payment.	True	URL for payment callback notification, POST request.
TC-MerchantId	Merchant ID.	True	Merchant ID (mchid) is the unique identifier for the merchant in the superapp's payment. All API calls must include this parameter for the superapp's payment to verify the merchant's identity. This ID is provided to the merchant upon successful registration.

TC-TradeType	Transaction type	True	JSAPI: Users pay via the mini program, which forward to the superapp's payment API.
TC-UserID	Superapp user login ID.	True	-
Authorization	Signature authentication information.	True	Refer to APIv3 Certificate Signature

Body

Name	Type	Required	Description
appid	string	True	Merchant's mini program appid, the unique identifier for the merchant on Super App as a Service (SAS). Ensure this mini program appid is bound to the merchant ID.
description	string	True	The product information description, which should be accurately provided and cannot exceed 127 characters.
out_trade_no	string	True	Internal order number in the merchant system, must be 6-32 characters long, can only include numbers, uppercase and lowercase letters, _-|*, and must be unique within the same merchant ID.
time_expire	string	True	The payment end time, which is the last time the user can complete the payment for this order, not the order closing time. After this time, the user will not be able to pay for the order. Format requirements: The payment end time must follow the RFC 3339 standard format: yyyy-MM-DDTHH:mm:ss+TIMEZONE. yyyy-MM-DD represents the date; T separates the date and time; HH:mm:ss represents the time; TIMEZONE represents the time zone (e.g., +08:00 for Beijing time).
attach	string	False	Custom data packet provided by the merchant when creating the order, not visible to the user, used to store merchant

			custom information related to the order, with a total length limit of 128 characters. This field will be returned to the merchant in the order query API and payment success callback notification.
amount	object	True	Order amount.
amount.total	int	True	Total order amount, integer (in cents).
amount.currency	string	False	Currency type, a three-letter code compliant with ISO 4217.
payer	object	True	Payer information.
payer.openid	string	True	A unique identifier for each user within a single merchant's mini program appid. The user's openid must be obtained before placing an order. For details, see Mini Program Login
detail	object	True	Product information.
detail.cost_price	int	False	Original price of the order.
detail.goods_detail	object	True	Product list.
detail.goods_detail.merchant_goods_id	string	True	Product code provide by the merchant, composed of one or more of the following: Half-width uppercase and lowercase letters, numbers, hyphens, and underscores.
detail.goods_detail.goods_name	string	False	Actual product name.
detail.goods_detail.quantity	int	True	Quantity of products purchased by the user.
detail.goods_detail.unit_price	int	True	The unit price of the product, integer (in cents).

Response parameters

Name	Type	Required	Description
returnCode	string	True	Response code, 0 indicates success.
returnMessage	string	False	Response information.

data	object	True	Response data.
data.prepayId	string	True	Unique order ID.
requestId	string	True	Request trace ID.

Business error codes

Error code	Error code value	Description
220000	MCH_NOT_EXISTS	Check if the merchant ID is correct.
220001	TRADE_ERROR	Transaction failure, check the details returned by the API.
220002	ORDER_NOT_EXIST	Check if the order has been placed successfully.
220003	APPID_MCHID_NOT_MATCH	Check if the merchant's mini program appid matches the merchant ID.

2.2 Mini program order payment callback

When a user successfully pays for an order using the regular payment function, the superapp will send a callback notification via a POST request to the TC-Payment-Callback described in the section 1.7, informing the merchant that the user has completed the payment.

Path: TC-Payment-Callback, described in the section 1.7

Method: POST

API description:

Mini program order payment callback.

Request parameters

Headers

Parameter	Description	Required	Description
TC-Serial	The callback URL of a successful or failed payment.	True	The merchant serial number, and the merchant payment public key ID on the superapp's payment (Merchant serial number, merchant certificate).
TC-Signature	Signature value for verification.	True	Refer to APIv3 Certificate Signature
TC-Timestamp	Timestamp for verification.	True	-

TC-Nonce	Random string for verification.	True	-
TC-Prepay-Id	The mini program order ID.	True	-

Body

Name	Type	Required	Description
id	string	True	Unique identifier for the callback notification.
create_time	string	True	Notification creation time: It follows the RFC 3339 standard format: yyyy-MM-DDTHH:mm:ss+TIMEZONE. yyyy-MM-DD represents the date; T separates the date and time; HH:mm:ss represents the time; TIMEZONE represents the time zone (e.g., +08:00 for Beijing time). Example: 2015-05-20T13:29:35+08:00 represents 13:29:35 on May 20, 2015 (Beijing time).
event_type	string	True	The superapp's payment callback notification types. Payment success - TRANSACTION.SUCCESS. Payment failure - TRANSACTION.PAYERROR.
resource_type	string	True	Type of notification data, fixed as encrypt-resource.
summary	string	True	Summary of the callback content from the superapp's payment.
resource	object	True	Notification data.
resource.algorithm	string	True	Encryption algorithm type: The encryption algorithm type for the callback data ciphertext. It is currently AEAD_AES_256_GCM. Developers need to use the same type of data for decryption.
resource.ciphertext	string	True	Data ciphertext: The Base64-encoded callback data ciphertext. The merchant is required to decode it using Base64 and decrypt it with the API key.
resource.associated_data	string	False	Additional data: Additional data involved in decryption. This field may be empty.
resource.original_type	string	True	Original callback type: The object type before encryption, which is transaction.

resource.nonce	string	True	Random string: The random string involved in decryption.

Response parameters

Name	Type	Required	Description
returnCode	string	True	Response code, 0 indicates success.
returnMessage	string	False	Response information.
data	Boolean	True	Indicates if the order was closed successfully.
requestId	string	True	Request trace ID.

2.3 Query orders by the order number

Path: /v3/pay/transactions/out-trade-no/{out_trade_no}

Method: GET

API description:

Queries orders by the merchant order number.

Request parameters

Headers

Parameter	Description	Required	Description
Authorization	Signature authentication information.	True	Refer to APIv3 Certificate Signature

Path

Parameter	Description	Required	Description
out_trade_no	Merchant order number.	True	-

Query

Parameter	Description	Required	Description
mchid	Merchant ID, provided when the merchant places the order.	True	-

Response parameters

Name	Type	Required	Description
returnCode	string	True	Response code, 0 indicates success.
returnMessage	string	False	Response information.
data	object	True	Response data.
data.app_id	string	True	The mini program appid, provided when the merchant places the order.
data.mchid	string	True	Merchant ID, provided when the merchant places the order.
data.out_trade_no	string	True	Merchant order number, provided when the merchant places the order.
data.transaction_id	string	True	Unique identifier of the order on the superapp's payment, returned after successful payment.
data.trade_type	string	False	Transaction type of the current order, possible values: JSAPI: Users pay via the mini program, which forward to the superapp's payment API.
data.trade_state	string	True	Transaction status, possible values: SUCCESS: Payment successful REFUND: Refund in process NOTPAY: Not paid CLOSED: Closed REVOKED: Revoked USERPAYING: Payment in process PAYERROR: Payment failed
data.trade_state_desc	string	True	A detailed description of the transaction status.
data.bank_type	string	False	Description of the user's payment method, returned after successful payment. Format: bank code_specific type (DEBIT debit card/CREDIT credit card).
data.success_time	string	False	The time when the user completes the payment for the order. This parameter is returned after the order is successfully paid. Format: Follow the RFC 3339 standard format: yyyy-MM-DDTHH:mm:ss+TIMEZONE. yyyy-MM-

			DD represents the date; T separates the date and time; HH:mm:ss represents the time; TIMEZONE represents the time zone (e.g., +08:00 for Beijing time). Example: 2015-05-20T13:29:35+08:00 represents 13:29:35 on May 20, 2015 (Beijing time).
data.payer	object	False	Information about the payer. The superapp user ID will be returned after the order is successfully paid.
data.amount	object	False	Order amount.
data.amount.total	string	False	The total amount of the order.
data.amount.payer_total	string	False	The actual amount paid by the user.
data.amount.currency	string	False	Currency type.
data.amount.payer_currency	string	False	The currency used by the user for the payment.
requestId	string	True	Request trace ID.

Business error codes

Error code	Error code value	Description
220000	MCH_NOT_EXISTS	Check if the merchant ID is correct.
220001	TRADE_ERROR	Transaction failure, check the details returned by the API.
220002	ORDER_NOT_EXIST	Check if the order has been placed successfully.

2.4 Close an order

Orders that are in an unpaid status can be closed using this API when payment is no longer required. Common scenarios for closing an order include:

The user submits a request to cancel the order in the merchant system, and the merchant needs to close the order.

The order has timed out without payment (exceeding the payment time set by the merchant system or the time_expire payment deadline set when placing the order), and the merchant needs to close the order.

Path: /v3/pay/transactions/out-trade-no/{out_trade_no}/close

Method: POST

API description:

Queries orders by the merchant order number.

Request parameters

Headers

Parameter	Description	Required	Description
Authorization	Signature authentication information.	True	Refer to APIv3 Certificate Signature

Path

Parameter	Description	Required	Description
out_trade_no	Merchant order number.	True	-

Body

Parameter	Description	Required	Description
mchid	Merchant ID, provided when the merchant places the order.	True	-

Response parameters

204 No Content

No content body

Business error codes

Error code	Error code value	Description
220000	MCH_NOT_EXISTS	Check if the merchant ID is correct.
220001	TRADE_ERROR	Transaction failure, check the details returned by the API.
220002	ORDER_NOT_EXIST	Check if the order has been placed successfully.

Common error codes

Error code	Error code value	Description
210000	PARAM_ERROR	Provide correct parameters according to the error prompt.
210001	INVALID_REQUEST	Check the API parameter rules.
210002	SIGN_ERROR	Incorrect signature.

210003	SYSTEM_ERROR	Try again later.

3. Mini game payment APIs

3.1 Mini game order creation API

Path:/v3/pay/transactions/mg/create-prepay-order

Method: POST

API description:

Creates a mini game order.

Request parameters

Parameter	Type	Description	Required	Description
AppId	string	Superapp ID	True	Superapp ID.
MiniAppId	string	Mini game ID	True	Mini game ID.
AccountId	string	Account ID	True	The mini game account ID.
SignData	string	Original payment string.	True	For specific payment parameters, see signData below. The da Example for signData: '{"mode":"goods","offerId":"123","buyQuantity":1,"env":0,"curre
PaySig	string	Payment signature.	True	The pay_sig parameter uses a signature algorithm that signs side payment module. The signature algorithm pseudocode is paySig = to_hex(hmac_sha256(appKey,'requestMidasPayme For details, see Mini Game Payment Signature and Verificati
Signature	string	User session signature.	True	User session signature. For details on signature parameter signature algorithm, see M
Lang	string	Language	True	zh_CN: Chinese; en: English

Response parameters

Name	Type	Required	Description
returnCode	string	True	Response code, 0 indicates success.

returnMessage	string	False	Response information.
data	object	True	Response data.
data.prepayId	string	True	Unique order ID.
requestId	string	True	Request trace ID.

Business error codes

Error code	Error code value	Description
1	-	System failure.
-2	-	Payment canceled.
-6	-	Incorrect order parameter type.
-15001	-	Virtual payment API error: Missing parameter.
-15002	-	Virtual payment API error: Invalid parameter.
-15003	-	Virtual payment API error: Duplicate order.
-15004	-	Virtual payment API error: Backend error.
-15005	-	Virtual payment API error: Superapp ID permission has been banned.
-15006		Virtual payment API error: Currency type not supported.
-15007	-	Virtual payment API error: Order already paid.
-15009	-	Virtual payment API error: Payment exceeds health system limit (a default popup will appear)
-15010	-	Virtual payment API error: Sandbox payment are not allowed for released mini games.
-15011	-	Invalid request data type.
-15012	-	Signature error.
-15013	-	Virtual currency not issued.
-15014	-	PaySig error.
-15015	-	Session key expired.

-15016	-	Virtual item price error.
-15017	-	Order closed.
1	-	Virtual payment API error: User cancelled payment.
2	-	Virtual payment API error: Client error, a new payment request was made while one was ongoing
3	-	Virtual payment API error: Android-specific error. User attempted to use Google Play for payment, but Google Play is not installed on the device.
4	-	Virtual payment API error: The user's operating system payment status is abnormal.
5	-	Virtual payment API error: Operating system error.
6	-	Virtual payment API error: Other errors.
7	-	Virtual payment API error: Payment canceled.
1000	-	Parameter error.
1001	-	Server region not released.
1003	-	Virtual currency not issued, merchant number banned, or Midas Portal error. Please ensure the virtual currency is issued and check if the merchant number is banned.
3017/-15012	-	Invalid virtual item ID.
701001	-	Payment is prohibited on iOS.

3.2 Mini game payment callback

Path:/super-app/transactions/mg/callback

Method: POST

API description:

Mini game payment callback

Request parameters

Parameter	Type	Description	Required	Description
EventType	string	Message type.	True	Message type. For virtual item delivery, this is always: event

Event	string	Event type.	True	Event type. For direct virtual item purchase (marketplace): minigame_h5_goods_deliver_notify For in-game direct purchase, this is always: minigame_game_pay_goods_deliver_notify
PayModel	string	Payment method	True	Payment method.
Payload	string	Payment payload data.	True	Detailed content in JSON format. See the Payload table below. (All message content is formatted as JSON for unified signature verification.)
PayEventSig	string	Payment signature.	True	Payment signature. For details, refer to Mini Game Payment Signature and Verification

Payload

Parameter	Type	Description	Required	Description
OpenId	string	openid	True	The openid of the player who receives the virtual item.
Env	string	Environment configuration	True	Environment configuration. 0: Live environment (also known as production environment) 1: Sandbox environment
OutTradeNo	string	Order number	True	Order number.
GoodsInfo	Object	The virtual item info	True	The virtual item info.

GoodsInfo

Parameter	Type	Description	Required	Description
ProductId	String	Game item ID identifier	True	Game item ID identifier.
Quantity	Number	Quantity of game items purchased	True	Quantity of game items purchased.
ZoneId	String	Server region ID	True	Server region ID.
OrigPrice	Number	Original game item	True	Original game item price (in

		price (in cents)		cents).
ActualPrice	Number	Actual paid price (in cents)	True	Actual paid price (in cents).
Attach	String	Pass-through data	True	Pass-through data.
OrderSource	Number	1 In-game order 2 Marketplace order 3 Marketplace test order	True	1 In-game order 2 Marketplace order 3 Marketplace test order

Response parameters

Name	Type	Required	Description
returnCode	string	True	Response code, 0 indicates success.
returnMessage	string	False	Response information.
data	string	True	Response data, OK indicates success.
requestId	string	True	Request trace ID.