**Playsout SDK Integration**

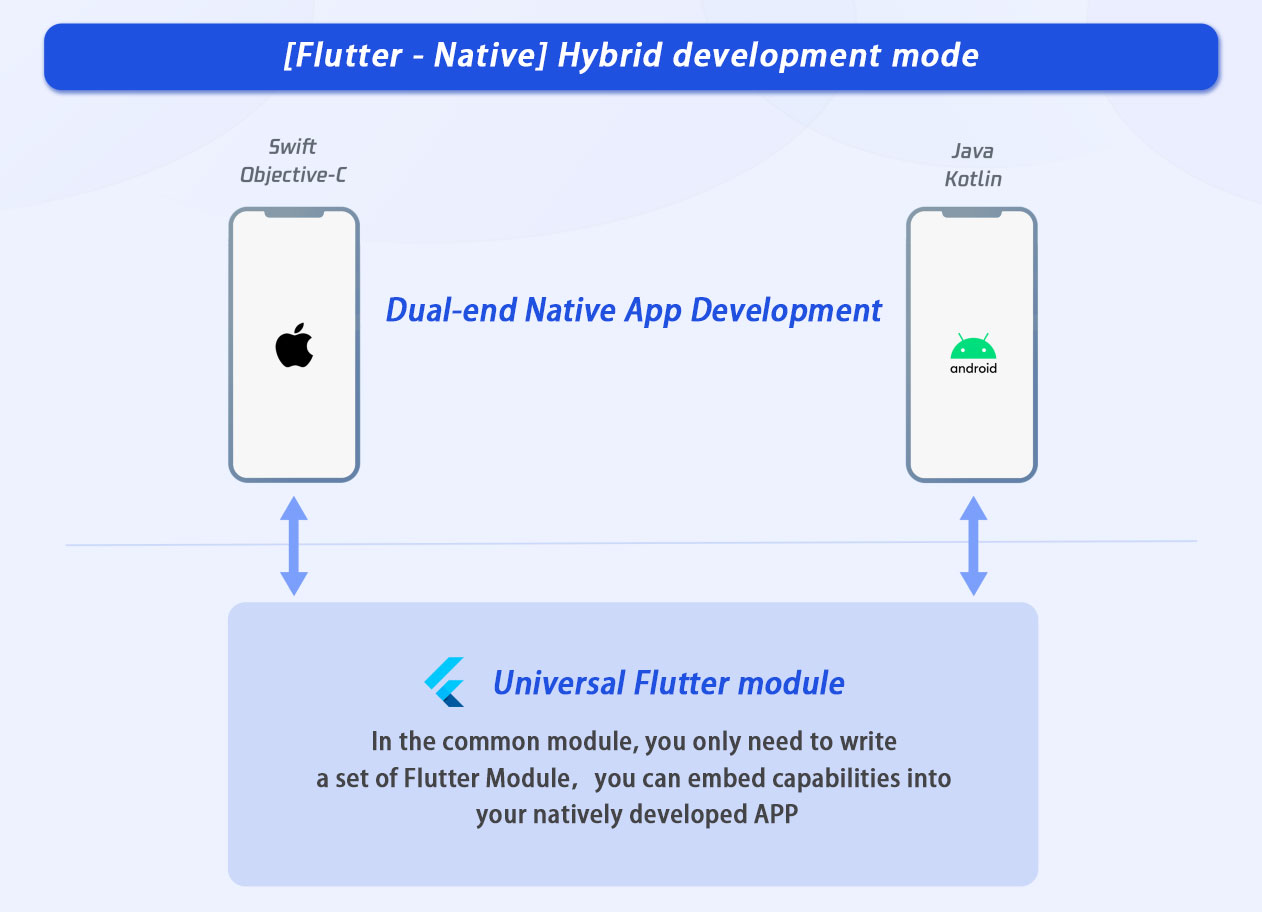
1. **Overview**

Playsout Mini Game Container SDK is developed using Flutter + C++ at the underlying layer, combining rapid cross-platform capabilities with high performance. By reading this document, you will learn how to integrate the Playsout Mini Game Container into your existing native Android/iOS development projects.

1. **Technical Requirements**

If you wish to incorporate Playsout Mini-Game Container capabilities into your existing App, we recommend adopting a hybrid development approach by embedding the Flutter module into your native App project.

This significantly reduces workload and enables rapid implementation of Playsout Mini-Game Container functionality in both Android and iOS native Apps.



## Environment Requirements

|  |  |
| --- | --- |
| Environment | Version |
| Android | Android Studio 3.5 or higher. The app requires devices running Android 5.0 or later. |
| iOS | Xcode 11.0 or higher. Ensure your project has a valid developer signature configured. |

**3. Usage on Native Platforms**

### **P****r****inciple – Flutter Hybrid Development**

The core principle is to package the Flutter module into an executable program for the native platform and embed it into the native project. Since the Flutter module is reusable, you only need to write it once and embed it into both Android and iOS apps.

When your existing app needs to display Playsout-related pages, load the corresponding Activity (Android) or ViewController (iOS) that hosts Flutter. For cross-platform communication (e.g., passing user information or other data), use the Method Channel.

1. **Adding the Flutter Module to an Android Project**

Playsout SDK provides an Android Archive (AAR). For specific integration steps, refer to:

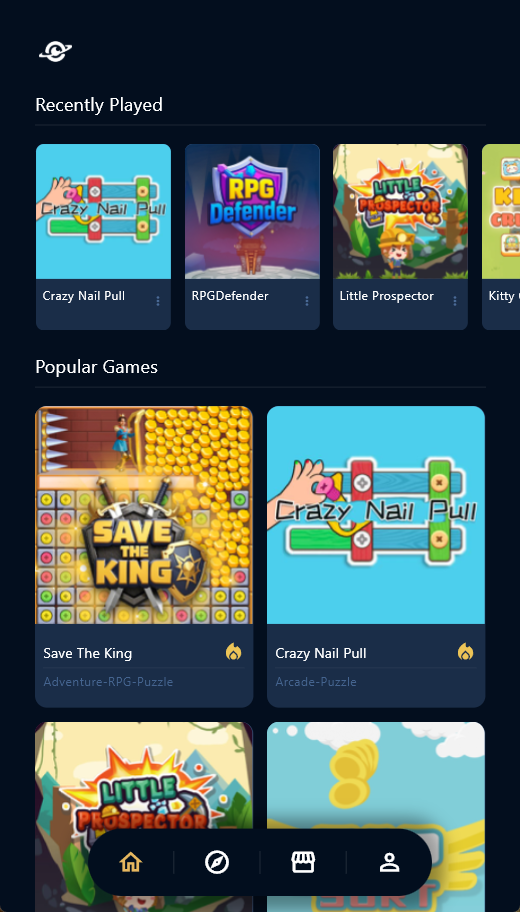
<https://docs.flutter.dev/add-to-app/android>

1. **Adding the Flutter Module to an iOS Project**

Playsout SDK provides Xcode frameworks. For specific integration steps, refer to: <https://docs.flutter.dev/add-to-app/ios>

**4. Integration**

This section explains how to quickly integrate the Playsout Mini Game component into an iOS project. You will complete the following key steps and ultimately implement a mini-game feature with a complete UI.

## **Environment Preparation**

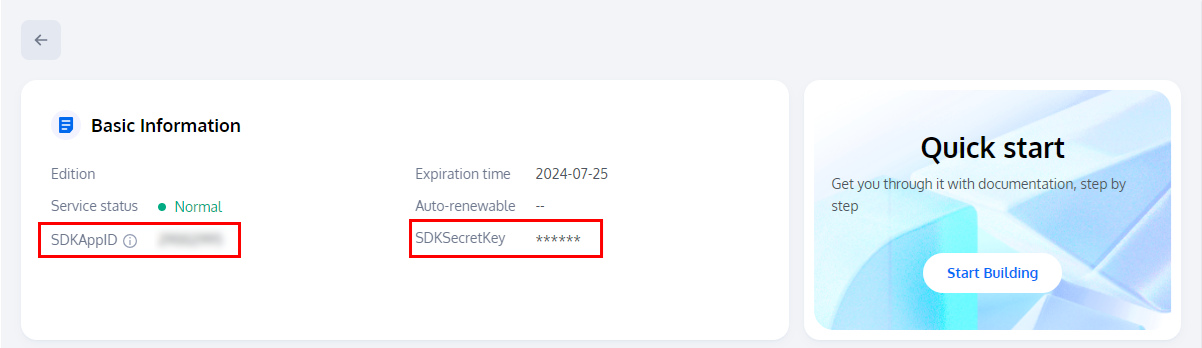
XCode 11.0 or later.

iOS 16 or later.

Physical device running iOS 16.0 or later.

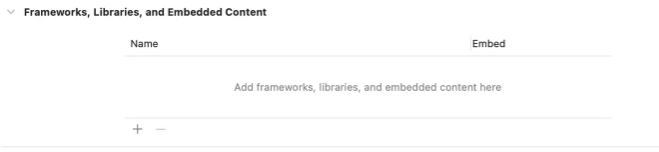
## **Enabling Services**

Before using the Playsout Mini Game Container service, you need to enable the service for your app(contact business partners to access the control platform). After enabling the service, note down the ChannelID and SDKKey, which will be used in subsequent steps. You can use test chanelID & SDKKey for testing now.

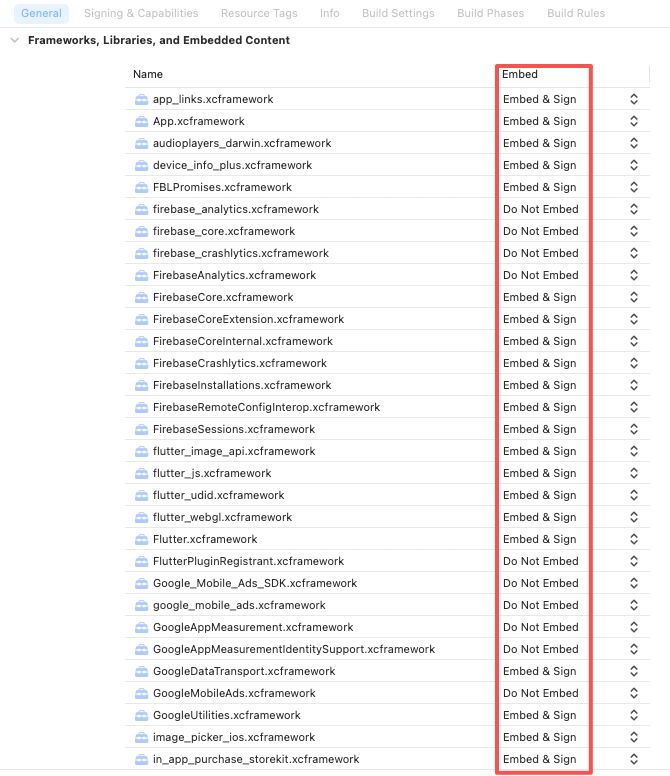


## **SDK Import and Configuration**

Download playsoutsdk(<https://github.com/Playsout/PlaysoutiOSSDK>), In the Xcode project, navigate to Target > General > Frameworks, Libraries, and Embedded Content . You can either: 1.Click the plus sign (+) to add these frameworks, or 2.Directly drag these frameworks into this section.



**Config frameworks referencing demo:**

****

Config your MyApp.xcodeproj

refer to:https://docs.flutter.cn/add-to-app/ios/project-setup

1.Navigate to the /path/to/MyApp/ directory.

2.Right-click on MyApp.xcodeproj and select Show Package Contents.

3.Open project.pbxproj with Xcode. The file opens in Xcode's text editor. This also locks Project Navigator until you close the text editor.

4.Find the lines that resemble the following text in the /\* *Begin PBXFileReference section* \*/.

312885572C1A441C009F74FF /\* *Flutter.xcframework* \*/ = {

isa = PBXFileReference;

expectedSignature = "AppleDeveloperProgram:S8QB4VV633:FLUTTER.IO LLC";

lastKnownFileType = wrapper.xcframework;

name = Flutter.xcframework;

path = Flutter/Release/Flutter.xcframework;

sourceTree = "<group>";

};

312885582C1A441C009F74FF /\* *App.xcframework* \*/ = {

isa = PBXFileReference;

lastKnownFileType = wrapper.xcframework;

name = App.xcframework;

path = Flutter/Release/App.xcframework;

sourceTree = "<group>";

};

5.Change the Release text highlighted in the prior step and change it to $(CONFIGURATION). Also wrap the path in quotation marks.

312885572C1A441C009F74FF /\* *Flutter.xcframework* \*/ = {

isa = PBXFileReference;

expectedSignature = "AppleDeveloperProgram:S8QB4VV633:FLUTTER.IO LLC";

lastKnownFileType = wrapper.xcframework;

name = Flutter.xcframework;

path = "Flutter/$(CONFIGURATION)/Flutter.xcframework";

sourceTree = "<group>";

};

312885582C1A441C009F74FF /\* *App.xcframework* \*/ = {

isa = PBXFileReference;

lastKnownFileType = wrapper.xcframework;

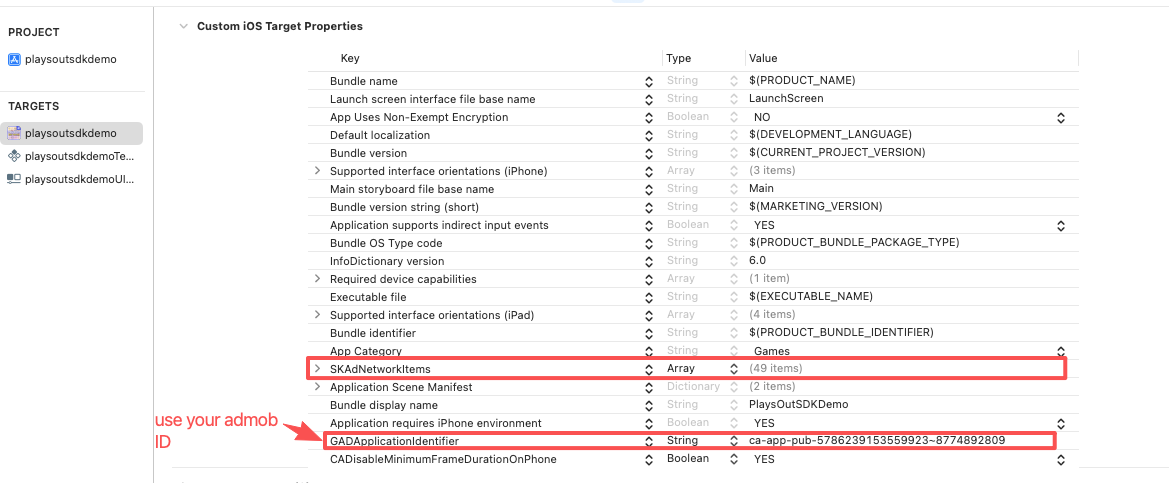
name = App.xcframework;

path = "Flutter/$(CONFIGURATION)/App.xcframework";

sourceTree = "<group>";

};

Add required configurations in Info.plist referencing demo:



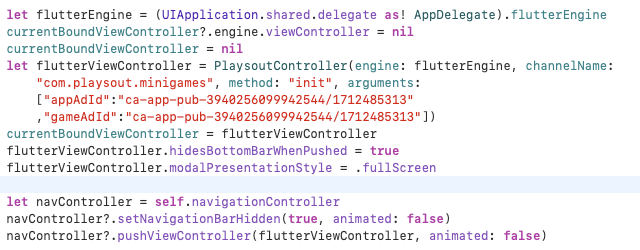
## **Initialize the Playsout SDK**

Add the following code to your project. Its purpose is to complete the Playsout SDK login by calling relevant interfaces. This step is critical—only after successful login can you use Playsout’s features.



## **Opening the Mini Game Library and Launching Your First Game**

To launch the mini-game container in your target ViewController, execute the following code:

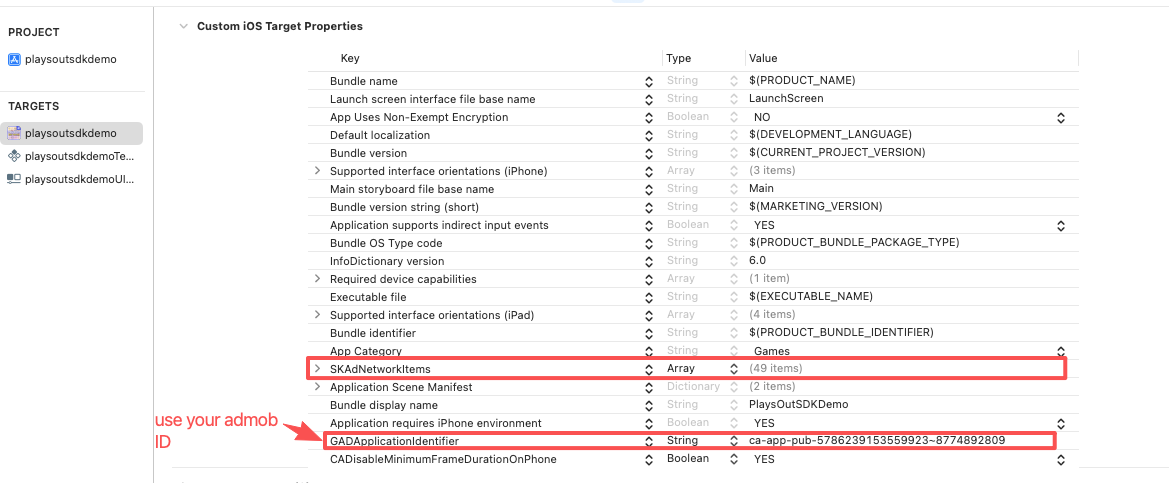


Use your admob AD ID to replace arguments(appAdId&gameAdId);"ca-app-pub-3940256099942544/1712485313" is test mode.

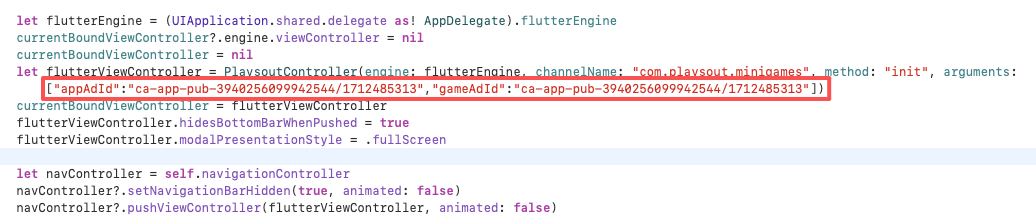
## 1744716360704 **1744716770138**

**6) How to add your admob AD**

1.Register with AdMob, add two rewarded ad uints. 2.In Info.plist, add configuration entries for GADApplicationIdentifierand SKAdNetworkItems.



3.Modify the code to use your rewarded ad ID.



**7) How to add your in app purchase**

In your App Store Connect, add the following consumable in-app purchase items to your app. 'com.playsout.minigames.gems12', US$0.99

'com.playsout.minigames.gems60', US$4.99

'com.playsout.minigames.gems180', US$14.99

'com.playsout.minigames.gems360', US$29.99

'com.playsout.minigames.gems600', US$49.99

'com.playsout.minigames.gems1200', US$99.99

