

Haptik Webkit Flutter Integration Guide

Quick Start

Prerequisites

To Integrate Haptik SDK, you must obtain the following credentials to get started

- Client ID: Unique Haptik identification number associated with your account
- Business ID: Unique business identification number
- Base URL: Link pointing to specific server environment (Different for Staging and Production)
- Auth Type: authentication type to be used for user registration
- Auth ID: authentication type to be used for user registration

Getting Started

1. Create a Flutter Project

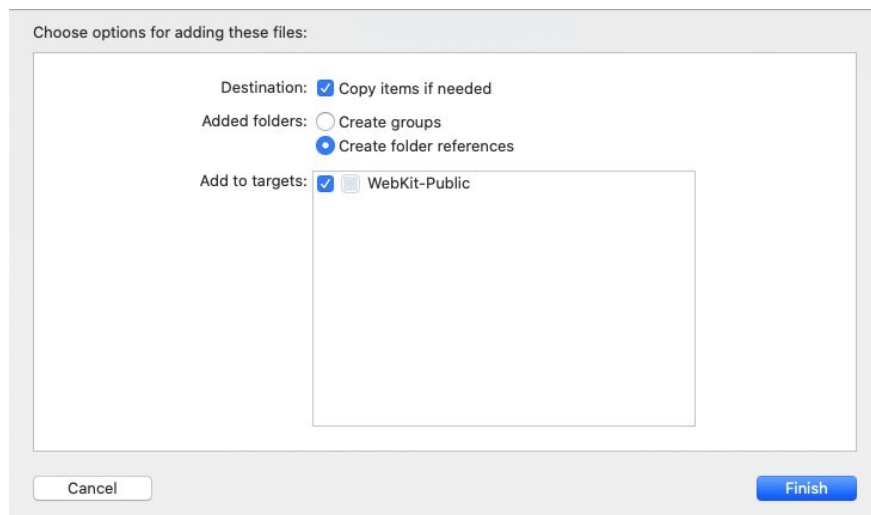
1. Create a new Flutter Project, if you have not already created one.
2. Once created, add a dependency on the “haptik_sdk” package in your “pubspec.yaml” file. You can either do this manually by navigating to the file and adding the dependency, or you can enter this command in your terminal “flutter pub add haptik_sdk”.
3. Your project is now ready to integrate the Haptik Webkit.

2. iOS Integration

1. **Install Haptik SDK** - For using the Haptik Webkit on iOS devices via Flutter, we will first have to install the Haptik SDK in our iOS files. To do this, you can open the iOS folder in Xcode.

- **Manual Installation**

1. Download the HPWebKit.zip file and unzip it.
2. Drag and drop the downloaded XCFramework into your Xcode project.
3. If the framework’s symbols are unable to be loaded, navigate to the “General” pane of your target and find the “Frameworks, Libraries, and Embedded Content” dropdown. Switch HPWebKit.xcframework from “Do Not Embed” to “Embed and Sign”



4. You will also need to install a HPWebKit dependency.

- **Auto Installation - CocoaPods**

1. If you haven't already done so, install a recent version of CocoaPods.
2. If you don't have an existing Podfile, run the following command to create one:



```
Terminal
$ pod init
```

3. Add this line to your Podfile:

```
use_frameworks!

target YourTargetName do pod 'HPWebKit'
end
```

4. Run the following command:



```
Terminal
$ pod install
```

2. **Setup Credentials & Environment** - To define your credentials for the iOS build of your application we need to define them in “info.plist” file. Define a dictionary with the key – “HaptikLib” and add in the following credentials you were given after registering as a Haptik partner:

- Client ID
- Base URL (Different for Staging and Production)
- Business ID

```
<key>HaptikLib</key>
<dict>
<key>baseUrl</key>
<string>insert_base_url</string>
<key>businessID</key>
<string>insert_busines_id</string>
<key>clientId</key>
<string>insert_client_id</string>
<key>runEnvironment</key>
<string>1</string>
</dict>
```

Note: While releasing the application on the App Store, only the Production URL (provided by a Haptik) should be used and the runEnvironment should be 0.

3. **Permissions** - HaptikLib requires some basic permissions which almost every application takes to function properly. You can see all the required permissions [here](#). Make sure to add them else iOS will assert the application when the SDK will try to present the prompt if the respective permission is not granted by the user.

4. **Initialize iOS SDK** – You need to initialize our Haptik SDK before you can use it. Head to the AppDelegate.swift file present in your Flutter Projects “iOS/Runner” directory.

```

1 HPKit.sharedSDK.setup()
2 let flutterViewController: FlutterViewController = window?.rootViewController as! FlutterViewController
3 let botChannel = FlutterMethodChannel(
4     name: "Haptik_FlutterAPP",
5     binaryMessenger: flutterViewController.binaryMessenger
6 )
7 let navigationController = UINavigationController(rootViewController: flutterViewController)
8
9 // navigationController.isNavigationBarHidden = true
10 window?.rootViewController = navigationController
11 window?.makeKeyAndVisible()

```

Add the above code under the “GeneratedPluginRegistrant.register(with: self)” line.

Here, we are creating a FlutterViewController that will be used by our Flutter App show the Conversation on a new screen.

Then we are creating a MethodChannel, which our Flutter code will use to communicate with the native iOS code and execute some functionality.

Finally, we are creating a UINavigationController which will be used to navigate to the FlutterViewController to display the conversation.

5. **Initialize Android SDK** – For android, we majorly make use of the haptik_sdk flutter package, so we do not have to initialize the android SDK. Instead, we have to increase our minSdkVersion to 21 in our android/app/build.gradle file. Secondly, The Haptik SDK will take in the required credentials in your flutter project through the android part. You'll have to add to in the strings.xml folder which resides in the following path: android/app/src/main/res/values If you don't have strings.xml in values folder, then you can even add the file yourself In strings.xml :

```

<resources>
  <string name="haptik_base_url">PROVIDED_BASE_URL</string>
  <string name="haptik_client_id">PROVIDED_CLIENT_ID</string>
  <string name="haptik_business_id">PROVIDED_BUSINESS_ID</string>
</resources>

```

These are the steps needed to initialize the SDK on both iOS and Android.

6. There are two ways to register a user.

- **Guest**
- **Custom Client Sign Up**

For you to take the user to the conversation screen where the user interacts/chats with the bot/agent, you need to provide user details to SDK

Guest: SDK creates a new user, without any specific details

Custom: For cases when the client/parent application already has a signup flow in place and wants to link the same user to the SDK, we support Custom Signups.

We will create 2 functions that will initiate the respective ways to register a user.

```
1 private func handleGuestBotLaunch(navigationController: UINavigationController) {
2     do {
3         try HPKit.sharedSDK.loadGuestConversation(
4             launchController: navigationController.topViewController!,
5             customData: nil
6         )
7     } catch {
8         print("Haptik Error : \(error)")
9     }
10 }
```

Guest Conversation (iOS)

```
1 private func handleCustomBotLaunch(args: [String : String], navigationController: UINavigationController) {
2
3     let authAttribute = HPAttributesBuilder.build { (builder) in
4         builder.authID = args["setAuthId"] ?? "NA"
5         builder.authCode = args["setAuthCode"] ?? "NA"
6         builder.userName = args["setUserName"] ?? "NA"
7         builder.email = args["setEmail"] ?? "NA"
8         builder.mobile = args["setMobile"] ?? "NA"
9         builder.signupType = args["setSignupType"] ?? "NA"
10    }
11
12    do {
13        try HPKit.sharedSDK.loadConversation(
14            launchController: navigationController.topViewController!,
15            attributes: authAttribute,
16            customData: nil
17        )
18        HPKit.sharedSDK.logout()
19    } catch {
20        print("Haptik Error : \(error)")
21    }
22 }
```

Custom Client Conversation (iOS)

To use these functions we need to handle our MethodChannel Calls. We will do this using setMethodCallHandler.

```

1  botChannel.setMethodCallHandler({
2    (call: FlutterMethodCall, result: FlutterResult) → Void in
3
4    switch call.method {
5      case "botCustomLaunch":
6        let args = call.arguments as? Dictionary<String, String>
7        handleCustomBotLaunch(
8          args: call.arguments as? Dictionary<String, String> ?? ["NA":"NA"],
9          navigationController: navigationController
10       )
11       break
12
13      case "botGuestLaunch":
14        handleGuestBotLaunch(navigationController: navigationController)
15       break
16
17      case "botClose":
18        navigationController.popViewController(animated: true)
19       break
20
21      default:
22        result(FlutterMethodNotImplemented)
23       break
24    }
25  })

```

Add this section of code, under the “window?.makeKeyAndVisible()” line in your iOS/Runner/AppDelegate.swift file.

For Android we directly use the HaptikSDK package and initiate the conversation using functions as follows:

```

1  androidCustomBot() async {
2      final haptikSdkPlugin = HaptikSdk();
3      final signup = SignupData();
4      signup.setAuthCode = "7ae2exxxx37xxxxx0d07f9xxxxx769xxx5";
5      signup.setAuthId = "test";
6      signup.setSignupType = "third_party";
7      signup.setCustomData = {
8          'userName': "WrapperUser24",
9          'email': "abc334@gmail.com",
10         'mobile': "9872635454",
11     };
12
13     await haptikSdkPlugin.launchCustomSignupConversation(signup).then((value) {
14         haptikSdkPlugin.logout();
15     });
16 }
17
18 androidGuestBot() async {
19     final haptikSdkPlugin = HaptikSdk();
20     final initData = InitData();
21
22     await haptikSdkPlugin.launchGuestConversation(initData).then((value) {
23         haptikSdkPlugin.logout();
24     });
25 }

```

Here, we have 2 functions, `androidCustomBot()` and `androidGuestBot()`. We simply call these on button taps to initialise the conversation. For custom client bots, we use `SignupData()` and set our required data as well as customdata accordingly. For guest bots, we just make use of the `InitData()` present within the `haptik_sdk` package.

In both the cases above, we call the `logout()` function once the bot is closed. This ensures that you are logged out once you exit.

Similarly, to launch our bot on iOS, we create their functions invoking the respective `MethodChannel` call.



```
1  iosCustomBot() async {  
2      const platform = MethodChannel('Haptik_FlutterAPP');  
3      await platform.invokeMethod('botCustomLaunch', {  
4          'setAuthCode': "7ae2exxxx37xxxxx0d07f9xxxxx769xxx5",  
5          'setAuthId': "test",  
6          'setSignupType': "third_party",  
7          'userName': "WrapperUser24",  
8          'email': "abc334@gmail.com",  
9          'mobile': "9872635454",  
10     });  
11 }  
12  
13 iosGuestBot() async {  
14     const platform = MethodChannel('Haptik_FlutterAPP');  
15     await platform.invokeMethod('botGuestLaunch');  
16 }
```

Here, we simply initialise our platform MethodChannel and invoke the methods we created mentioned in this documentation above. For sending **custom data in iOS**, please refer to [this](#).