OneApp Quick Start Guide

This document explains the process of connecting your existing mobile app to the Intellipay OneApp for your selected device.

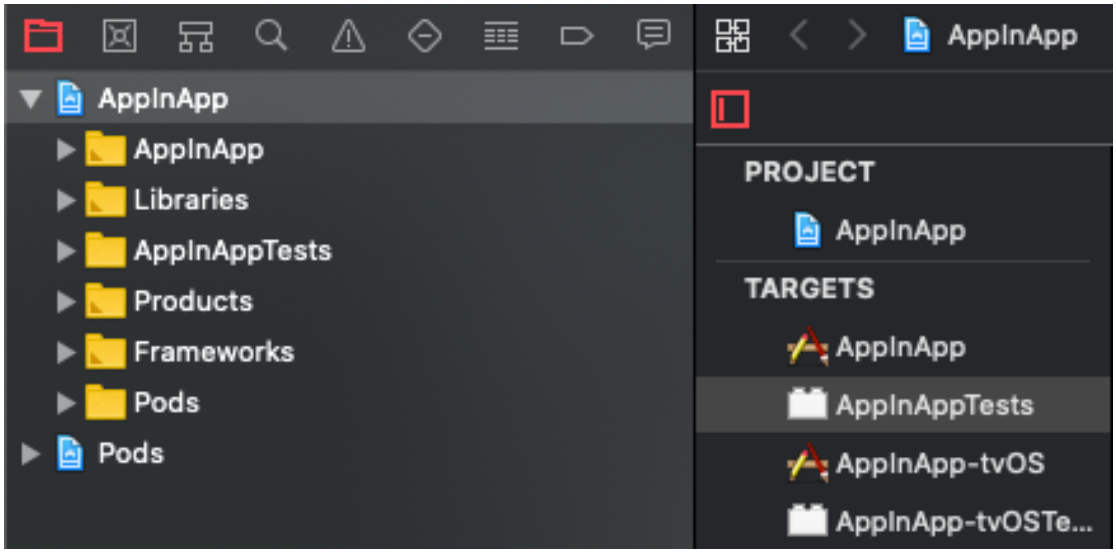
Intellipay's OneApp mobile solution allows you to easily make a payment through your own app. To enable this feature within your own app involves three parts. The first is to download the Intellipay app from the iOS App Store or Google Play Store. The second part is to configure your own application to open a deep link into our Intellipay app along with the appropriate parameters. The third part is to interpret the response back from our app so you can track (or at least display) the status on your end.

Table of Contents

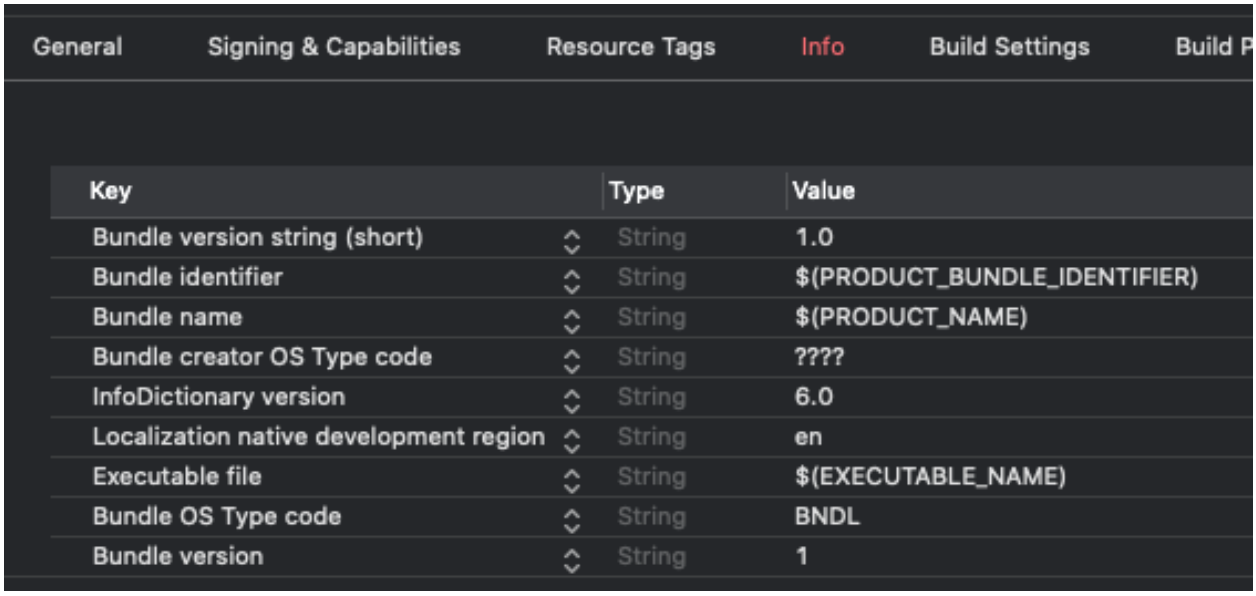
Deep Linking in iOS

STEP 1 – Configure URI Scheme

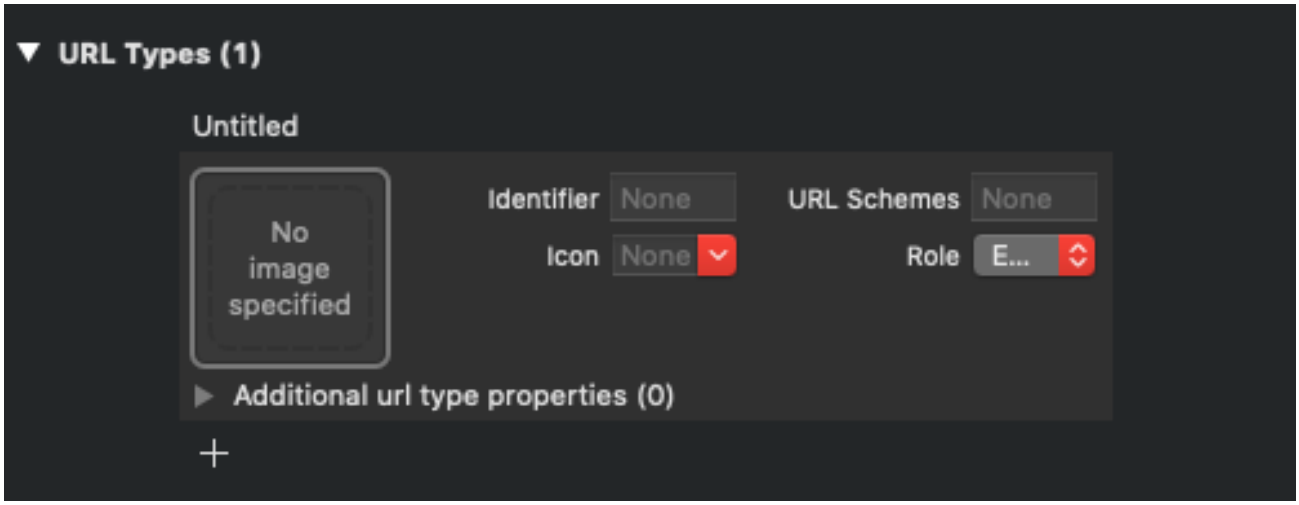
Open your project in Xcode and select the app from the left navigation bar.



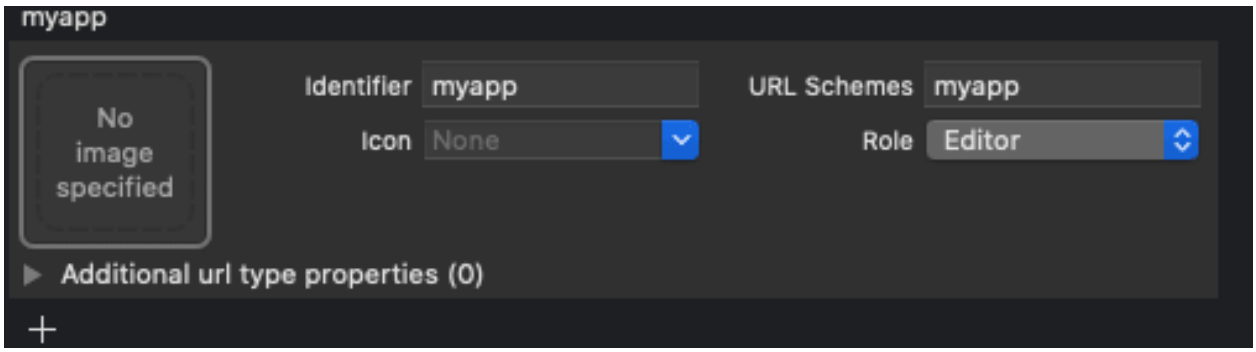
Open the Info tab.



Next, go to the URL Types section.



Click the + button and in the Identifier and Url Schemes inputs add your deep link url scheme (This can be anything you want it to be).



STEP 2: Setup deep link with parameters to pass to Intellipay App

SWIFT EXAMPLE

```
    @IBAction func onPressPayNow(_ sender: Any) {

let dic = [
  "amount": <PAYMENT_AMOUNT>,
  "apikey": <YOUR_INTELLIPAY_API_KEY>,
  "merchantkey": <YOUR_INTELLIPAY_MERCHANT_KEY>,
  "invoicenumber": <YOUR_PAYMENT_ID>,
  "customerid": <YOUR_CUSTOMER_ID>,
  "url": <YOUR_DEEP_LINK_URL>,
  "customername": <CUSTOMER_NAME>,
  "receiptemail": <CUSTOMER_EMAIL>
```

```
    ]

    var params = ""
    let encoder = JSONEncoder()
    if let paramsJSON = try? encoder.encode(dic) {
        if let jsonString = String(data: paramsJSON, encoding: .utf8) {
            print(jsonString)
            params = jsonString.addingPercentEncoding(withAllowedCharacters:
                .urlFragmentAllowed) ?? "{}"
        }

    }

    let application = UIApplication.shared;

    let intellipayAppPath = "intellipay://appinapp/\(params)"

    let appUrl = URL(string: intelliayAppPath)!


    if application.canOpenURL(appUrl) {
        application.open(appUrl, options: [:], completionHandler: nil)
    } else {
        print("can't open this url")
    }
}
```

Deep Linking in Android

STEP 1 - Configure URI Scheme

Open your projects AndroidManifest.xml file.

Set launchMode of MainActivity to singleTask in order to receive intent on existing MainActivity.

Add a new intent-filter inside of the MainActivity entry with a VIEW type action.

```
<activity
   android:name=".MainActivity"
   android:launchMode="singleTask">
   <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
   <intent-filter>
      <action android:name="android.intent.action.VIEW" />
      <category android:name="android.intent.category.DEFAULT" />
      <category android:name="android.intent.category.BROWSABLE" />
      <data android:scheme="myapp" />
   </intent-filter>
</activity>
```

STEP 2 - Setup deep link with parameters to pass to Intellipay App

Kotlin Example

```
fun makePayment(view: View) {

   val urlJson = JSONObject()
   urlJson.put("amount", <PAYMENT_AMOUNT>)
   urlJson.put("apikey", <INTELLIPAY_API_KEY>)
   urlJson.put("merchantkey", <MERCHANT_KEY>)
   urlJson.put("invoicenumber", <PAYMENT_ID>)
   urlJson.put("customerid", <CUSTOMER_ID>)
   urlJson.put("customername",<CUSTOMER_NAME>)
   urlJson.put("receiptemail", <CUSTOMER_EMAIL>)
   urlJson.put("url", <YOUR_DEEP_LINK_URL>)
```

```
    val URLparam = java.net.URLEncoder.encode(
        urlJson.toString(),
        "utf-8")

    val intent =        Uri.parse("intellipay://appinapp/$URLparam").let { app ->
        Intent(Intent.ACTION_VIEW, app)
    }

    startActivity(intent)
}
```

DEEP LINKING CALL PARAMETERS

These are the parameters that must be passed through the deep link URI. All parameters are in string format.

Parameter Name	Format or Example	Description
apikey	Ex: N34ASDFK342	This is the Intellipay apikey assigned to you
merchantkey	Ex: 8DFK34	This is the Intellipay merchantkey assigned to you
amount	Ex: 12.34	The amount you are charging your customer
invoicenumber	Ex: 123456	This is the id that you are using to identify this bill or invoice
customerid	Ex: 2344356	The id that you use to identify your customer
url	Ex: myapp	This is the deep link url scheme that you have defined for your app. This enables the return to your own application once you have completed a payment on the Intellipay app.
customername	Ex: John Doe	Name of customer or company
receiptemail	Ex: testemail@gmail.com	Email where you wish to send a receipt

RESPONSE HANDLING IN iOS

When you click the “Back to my app” button in the Intellipay app it will redirect you back to your app through the deep link URI that you sent in the “url” field in the json string of parameters. Here is an example of how to receive the response from the Intellipay app.

iOS Examples (Swift)

For iOS 13

Add the following to your “SceneDelegate.swift” file

```
func scene(_ scene: UIScene, openURLContexts URLContexts: Set<UIOpenURLContext>) {
      if let url = URLContexts.first?.url{
         // "url" contains the payment response
         // Handle the response however you see fit
         print(url)

      }
   }
```

For iOS 10

```
func application(_ app: UIApplication, open url: URL, options:
[UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
   print(url)
   return true
}
```

RESPONSE HANDLING IN ANDROID

When you click the “Back to my app” button in the Intellipay app it will redirect you back to your app through the deep link URI that you sent in the “url” field in the json string of parameters. Here is an example of how to receive the response from the Intellipay app.

Android Example (Kotlin)

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    // STEP 1 - Watch for new intent and trigger handleIntent function
    public override fun onNewIntent(intent: Intent) {
        this.intent = intent

        handleIntent()
    }

    // STEP 2 - Grab data from Uri sent from Intellipay app
    fun handleIntent() {
        // Get data from deep link uri
        val data: Uri? = intent?.data
        // This data will come in the following example's format
        // Example:
appinapp://{"status":28041562,"custid":22390060,"paymentid":28041562, ...}
        // The uri has one parameter which is a JSON string
    }

    // STEP 3 - Handle Response
    //  It is entirely up to you how you would like handle the response
    //  You will need to parse the JSON string from the Uri to get the
    //  different properties from the response
}
```