Red Hat Mobile Application Platform 4.4

Client SDK

For Red Hat Mobile Application Platform 4.4

Last Updated: 2018-10-08

Red Hat Mobile Application Platform 4.4 Client SDK

For Red Hat Mobile Application Platform 4.4

Legal Notice

Abstract

This document provides information on usage of RHMAP Client SDKs for all supported platforms.

Table of Contents

CHAPTER 1. NATIVE ANDROID

API Documentation

- RHMAP Client API - documentation for all RHMAP Client APIs
- RHMAP Android SDK Reference - generated Javadoc documentation of classes and methods of the RHMAP Android SDK

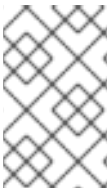
1.1. GRADLE

Gradle is a Java-based dependency management and build tool used for Android app development. It is used to distribute and integrate the RHMAP Native Android SDK with your app project.

1.1.1. Installing and Setting Up Gradle

Gradle requires Java JDK or JRE version 7 or higher. To check the version of Java you are currently using, execute the following command:

```
java -version
```



NOTE

Gradle uses the JDK version specified in your **PATH** environmental variable. You can also set the **JAVA_HOME** environmental variable to point to the installation directory of the desired JDK version.

It is recommended that you use Gradle version 2.4 or higher

To install Gradle, follow the installation instructions and setup instructions on the Gradle website.

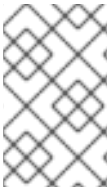
Alternatively, to install Gradle using the SDK manager tool, follow the section with instructions on installing Gradle using SDKMAN on the Gradle website.

1.2. MAVEN

Apache Maven is a project management and build automation tool for maintaining and distributing Java code. It is distributed under the Apache License, version 2.0. In an RHMAP context, is used to distribute the RHMAP Native Android SDK.

Maven 3.3 requires JDK 1.7 or later to execute. To check the version of Java you are currently using, execute the following command:

```
java -version
```



NOTE

Maven uses the JDK version specified in your **PATH** environmental variable. You can also set the **JAVA_HOME** environmental variable to point to the installation directory of the desired JDK version.

You can download the latest version of Maven from the Maven developers' website.

See the Maven documentation for detailed instructions on how to install and configure Maven.

1.3. GETTING STARTED

This SDK allows the user to use RHMAP APIs with native Android apps up to Android M (version 6.0, API level 23).

The RHMAP Android SDK is an open source project hosted in the FeedHenry Android SDK repository on Github. Feel free to fork it and contribute.

Before using the SDK, make sure you have the Android SDK or the Android Studio installed. You can download both from the Android Developer portal.

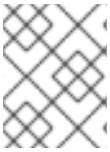
1.3.1. New App

Download the sample app to get started with a new native Android application which has the RHMAP SDK already included.

1.3.2. Existing App

Declare the following dependency in your **`build.gradle`** file:

```
compile 'com.feedhenry:fh-android-sdk:3.2.0'
```



NOTE

Add this dependency to the **app's `build.gradle`** file, not the **project's `build.gradle`** file.

If you are not using Gradle, you can manually download the SDK in a JAR along with its dependencies. For more details, see the Usage section in the README file of the SDK's Github repository.

If you use Maven to manage dependencies for your project, declare the **`fh-android-sdk`** as a dependency by adding the following code to the **`pom.xml`** file:

```
<dependencies>
  <dependency>
   <groupId>com.feedhenry</groupId>
   <artifactId>fh-android-sdk</artifactId>
   <version>3.2.0</version>
  </dependency>
</dependencies>
```

1.3.3. RHMAP Android Permission Setup

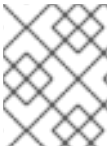
The SDK requires the **`INTERNET`** permission. If not already added, add the permission to the **`AndroidManifest.xml`** file:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

1.3.4. RHMAP Server Connection Setup

Define the properties which allow the app to communicate with RHMAP servers. Create a new file in the **`assets`** folder called **`fhconfig.properties`**. Add the following contents and replace the references, including the brackets, with the values from your project:

```
host = <RHMAP Core host>
projectid = <Project ID>
connectiontag = <Connection tag>
appid = <Client App ID>
appkey = <Client App API key>
```



NOTE

This data is not considered security sensitive data. For more on app security such as User Authentication, see $fh.auth Client API.

An example of a configured **`fhconfig.properties`** file:

```
host = https://cldapps.testsite.com
projectid = 0BiqQ5g8kfnUw8d1fn9NnAeR
connectiontag = 0.0.2
appid = 0BiqQAqVu9quH6BIrlH_LT9c
appkey = bfb20ee69d29add08e974eff89b9a571bacb7959
```

Additional properties you might encounter when working with RHMAP:

```
PUSH_SERVER_URL - <The hostname of the MBaaS target you will be targeting>
PUSH_SENDER_ID - <Firebase Sender ID>
PUSH_VARIANT- <UPS VARIANT>
PUSH_SECRET - <UPS SECRET>
```

See Projects - Connections for more information on connections between Client Apps and Cloud Apps.

1.3.5. Initialization

Before invoking any cloud requests from the RHMAP SDK, initialisation is required:

```
FH.init(this, new FHActCallback() {
  public void success(FHResponse pRes) {
    // Init okay, free to use FHActRequest
  }

  public void fail(FHResponse pRes) {
    // Init failed
    Log.e("FHInit", pRes.getErrorMessage(), pRes.getError());
  }
});
```

1.4. COMPATIBILITY

If the app targets Android M (API Level 23), use a version 3.x or higher of the RHMAP Android SDK,

ideally the latest version. The 3.x version of the SDK was introduced primarily because of a significant change in the Android SDK - the Apache HTTP Client. Apache HTTP Client (**`org.apache.http`** package) is no longer distributed with the Android SDK as of Android M. The RHMAP SDK now uses **`cz.msebera.android:httpclient`** as an HTTP client instead.

Upgrading an existing app to target Android M will also require an update to the app so it will use the new HTTP client. If Gradle is not being used, use the HTTP client's JAR files which are distributed with the RHMAP SDK and are available in the **`deps`** folder of the repository.

For information on compatibility with Android N (API Level 24), see Android N.

1.5. ANDROID N

Android N allows you to use certain Java 8 language features, such as lambda expressions. A lambda expression enhances the readability of code and also allows you to express instances of single-method classes more compactly. This contrasts with Anonymous classes which can be cumbersome. For more info, please refer to the lambda expression tutorial : https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html.

Within the context of RHMAP, lambda expressions have an impact on apps that use Jack and Java 8, specifically around the consumption of a lambda expression by the RHMAP APIs. With Java 8, a lambda expression can be passed into a method that consumes an Anonymous Inner class and it is likely that Lambda Expressions will impact DefaultCallbacks as shown in the following code:

```
@FunctionalInterface
public interface DefaultCallback extends FHActCallback {

    default void fail(FHResponse pResponse) {
        Log.e("DefaultCallback", pResponse.getErrorMessage());
    };

    @Override
    void success(FHResponse pResponse);
}
```

The following code snippet illustrates the use of a Lambda Expression within an RHMAP context:

```
@Override
protected void onStart() {
    super.onStart();
    FH.init(this, (DefaultCallback)(ignore) - > {

findViewById(R.id.check_online).setVisibility(View.VISIBLE);findViewById(R
.id.check_online).setOnClickListener(view - >
Toast.makeText(getBaseContext(), FH.isOnline() ? "Is online" : "Is not
online", Toast.LENGTH_LONG).show());
        }
    );
}
```

1.5.1. Migrating Apps to Android N

To migrate your app to Android N:

1. Make sure your code is working before migrating.
2. Update the fh-android-sdk dependency from **`3.1.0`** to **`3.3.0`** in your package management software.
3. Update the build tools and target APIs in the **`AndroidManifest`** and **`build.gradle`** files to API level 24.
4. Build your app locally and validate that everything is working.

CHAPTER 2. NATIVE IOS (OBJECTIVE-C)

Download

- SDK
- Sample App

API Documentation

- RHMAP Client API - documentation for all RHMAP Client APIs
- RHMAP iOS SDK Reference - generated documentation of classes and methods of the RHMAP iOS SDK

2.1. COCOAPODS

NOTE

CocoaPods is available only on OSX.

CocoaPods is a Swift And Objective-C dependency manager for Xcode projects. It is built with Ruby, which comes pre-installed with OSX. It is used to distribute the RHMAP iOS Swift and iOS Objective-C Client SDKs.

2.1.1. Install CocoaPods Using RubyGems

To install CocoaPods using the RubyGems package manager, execute the following command. This requires the **sudo** command to be enabled in OSX.

```
sudo gem install cocoapods
```

As an optional part of the CocoaPods setup, you can store your Podspec metadata locally at **~/.cocoapods/repos**. This helps to increase the dependency resolution speed and shortens the build time for your apps. To clone the spec-repo and create the directory, execute the following command:

```
pod setup
```

2.1.2. Install CocoaPods Without Using sudo.

Alternatively, to install CocoaPods without using sudo, follow the *Sudo-less Installation* section of the CocoaPods Getting Started Guide.

2.1.3. Install the Required Plugins

The RHMAP iOS Objective-C and Swift SDK packages rely on the **cocoapods-packager** and **cocapods-appledoc** plugins. To install both plugins, execute the following command:

```
[sudo] gem install cocoapods-packager cocoapods-appledoc
```

2.1.4. Enable CocoaPods in an Xcode Project

To enable Cocoapods Xcode app project after installation execute the following commands:

1. To navigate to the folder of your Xcode project, use:

```
cd <project_directory>
```

2. To create a podfile (if it does not already exist) in your project folder and automatically populate it with targets specified within the project, execute the following command:

```
pod init
```

2.1.5. Install Dependencies Using CocoaPods

To install the dependencies defined in the podfile of Xcode project using CocoaPods, execute the following command:

```
pod install
```

2.2. GET STARTED

This SDK lets you use RHMAP APIs in Objective-C apps for all iOS versions supported by RHMAP.

The RHMAP iOS Objective-C SDK is an open-source project hosted in the FeedHenry iOS SDK repository on Github. Feel free to fork it and make a contribution to this project.

Before using this SDK, make sure you have the Xcode IDE installed. You can download Xcode from the Apple Developer Portal.

If you plan to use CocoaPods to manage dependencies in your project, you must install it first:

```
sudo gem install cocoapods
```

2.2.1. New App

When starting with a new template app, you have two options:

- either use CocoaPods (the recommended approach and the default in Studio)
- or use FH framework.

2.2.1.1. Using CocoaPods

Clone the sample app to get started with a new iOS application which has the RHMAP SDK included as CocoaPods dependency.

```
git clone https://github.com/feedhenry-templates/blank-ios-app.git
cd blank-ios-app
```

Fetch the dependencies defined in the Podfile:

```
pod install
```

Open the **`blank-ios-app.xcworkspace`** workspace in Xcode. The required dependencies are located in the Pods group.

2.2.1.2. Using FH Framework

Clone the sample app to get started with a new iOS application which has the RHMAP SDK already included.

```
git clone https://github.com/feedhenry-templates/blank-ios-app.git
```

Open the **`blank-ios-app.xcworkspace`** workspace in Xcode.

2.2.2. Existing App

When you are integrating an existing app with RHMAP, you have two options:

- if your app uses CocoaPods, add the FH SDK as a new pod,
- otherwise add the FH framework to your Xcode project.

2.2.2.1. Using CocoaPods

Open the Podfile and add the following dependency:

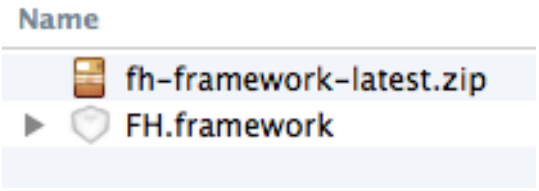
```
pod FH, '3.1.1'
```

Replace **`3.1.1`** with the version of RHMAP iOS Objective-C SDK you are targeting. If you do not specify a version number, the latest version in the CocoaPods central repository will be used.

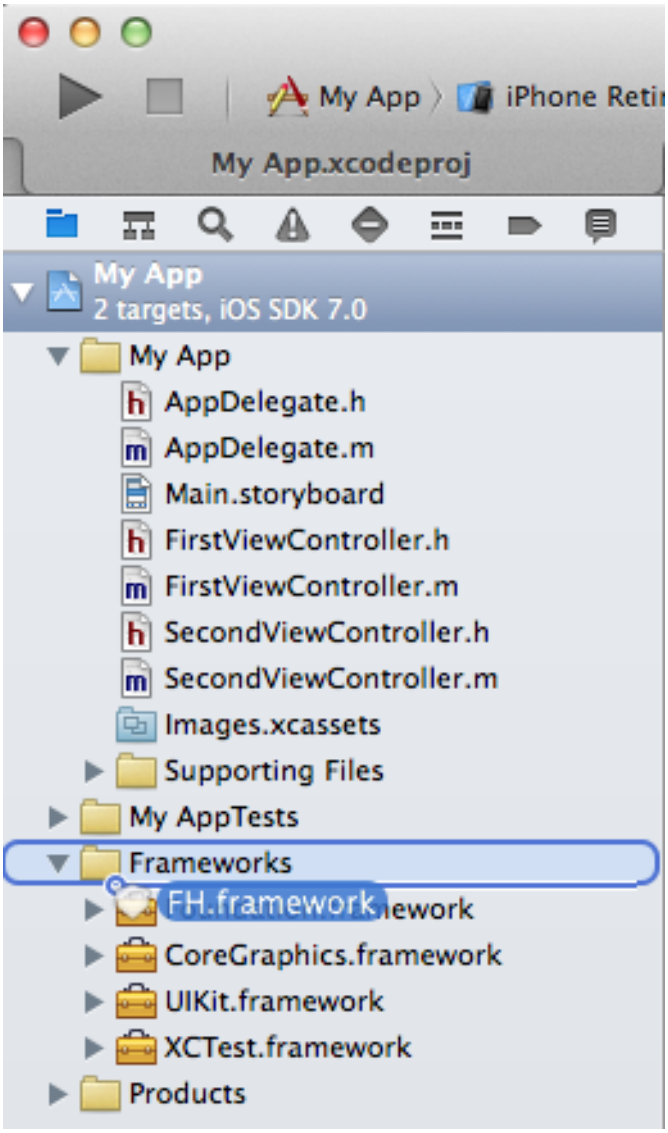
2.2.2.2. Using FH Framework

Download the SDK and save it as **`fh-framework-latest.zip`**.

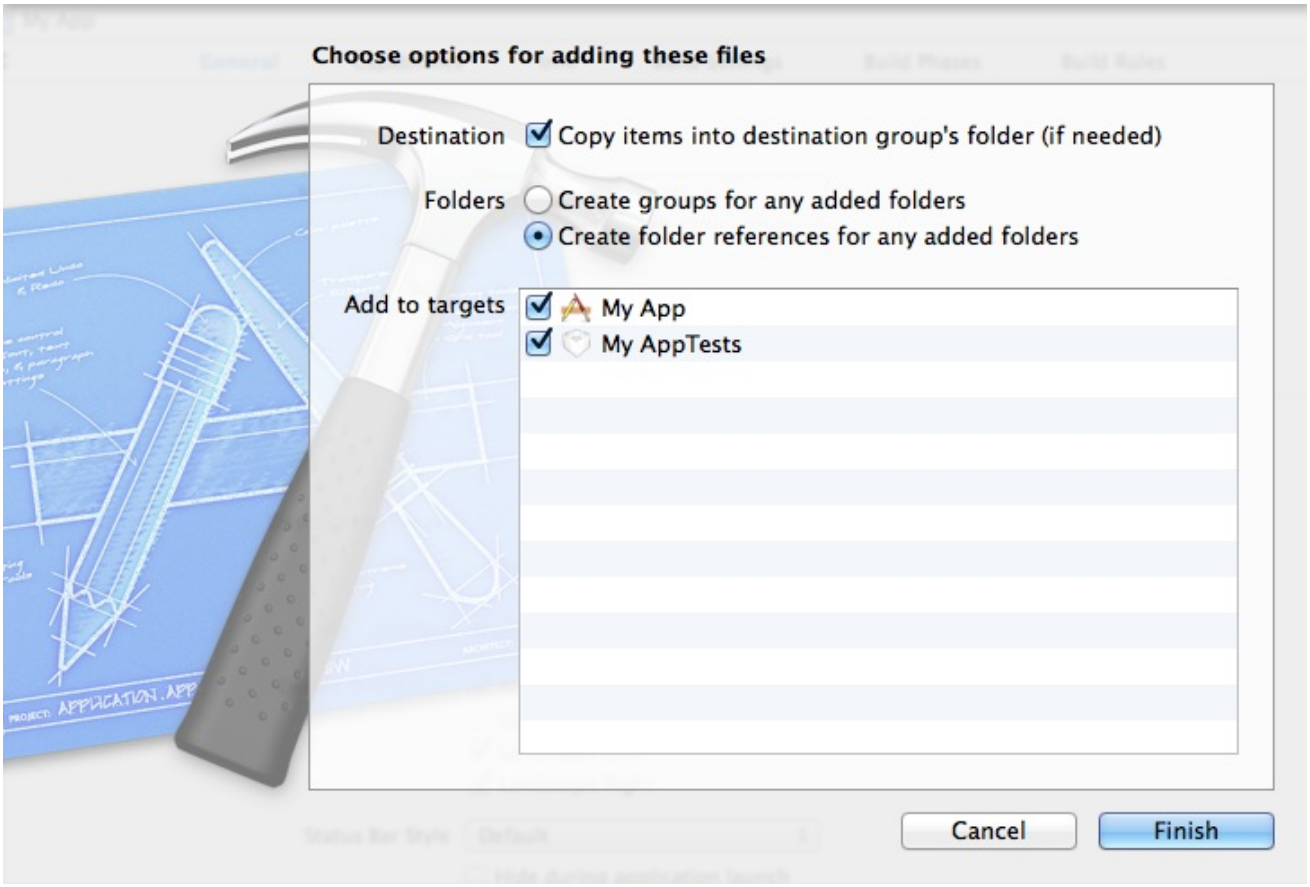
To integrate the SDK, extract the **`FH.framework`** file from the Zip file you downloaded above.



Once downloaded, drag the **`FH.framework`** file to your Xcode Project.



In the dialog, accept the defaults and click *Finish*.

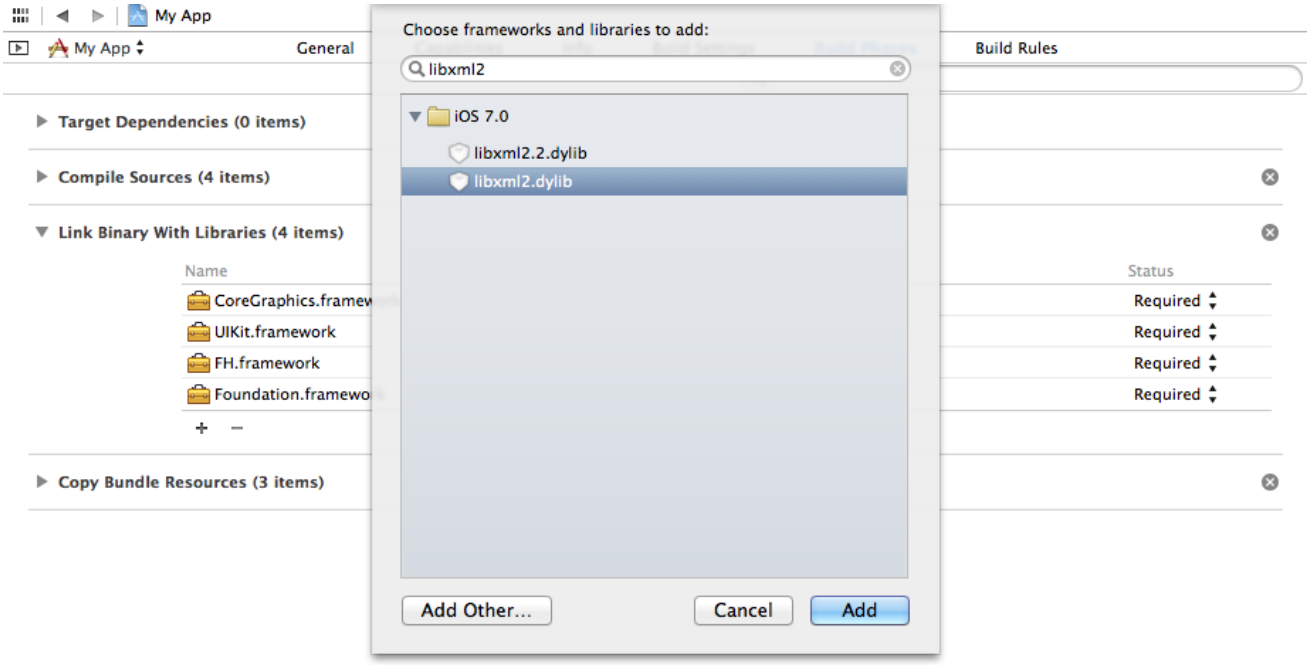


2.2.3. Integration

The RHMAP iOS SDK has a number of framework and library dependencies — you must re-configure your Build Phases to link against these frameworks and libraries at build time for the SDK to compile in your project correctly. The libraries you must link against are:

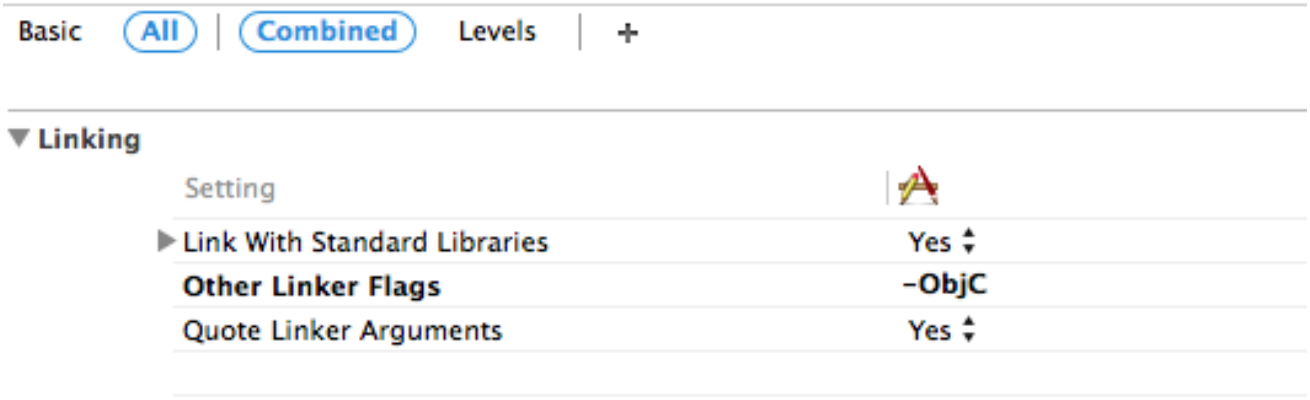
- **`libxml2.dylib`**
- **`libz.dylib`**
- **`SystemConfiguration.framework`**
- **`CFNetwork.framework`**
- **`MobileCoreServices.framework`**

To add these, go to **Build Phases > Link Binary with Libraries** and add the above dependencies as link dependencies.

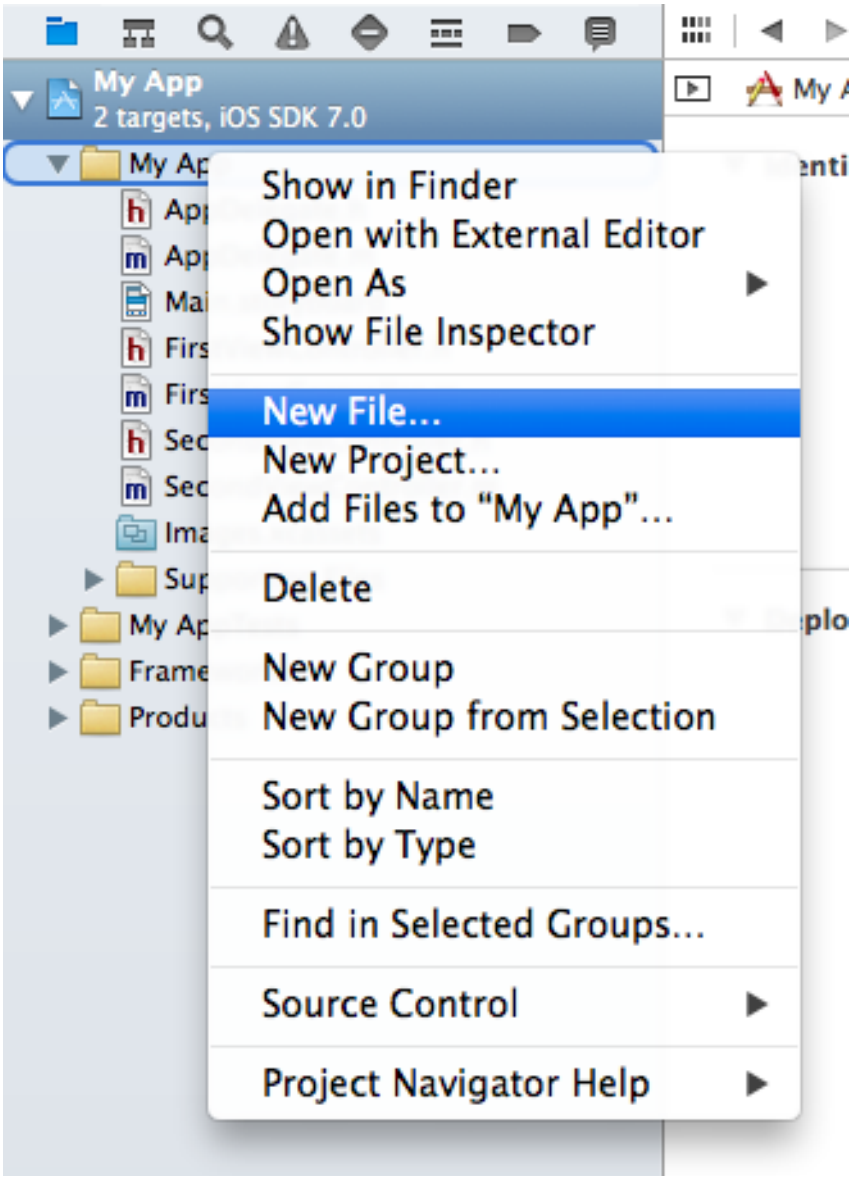


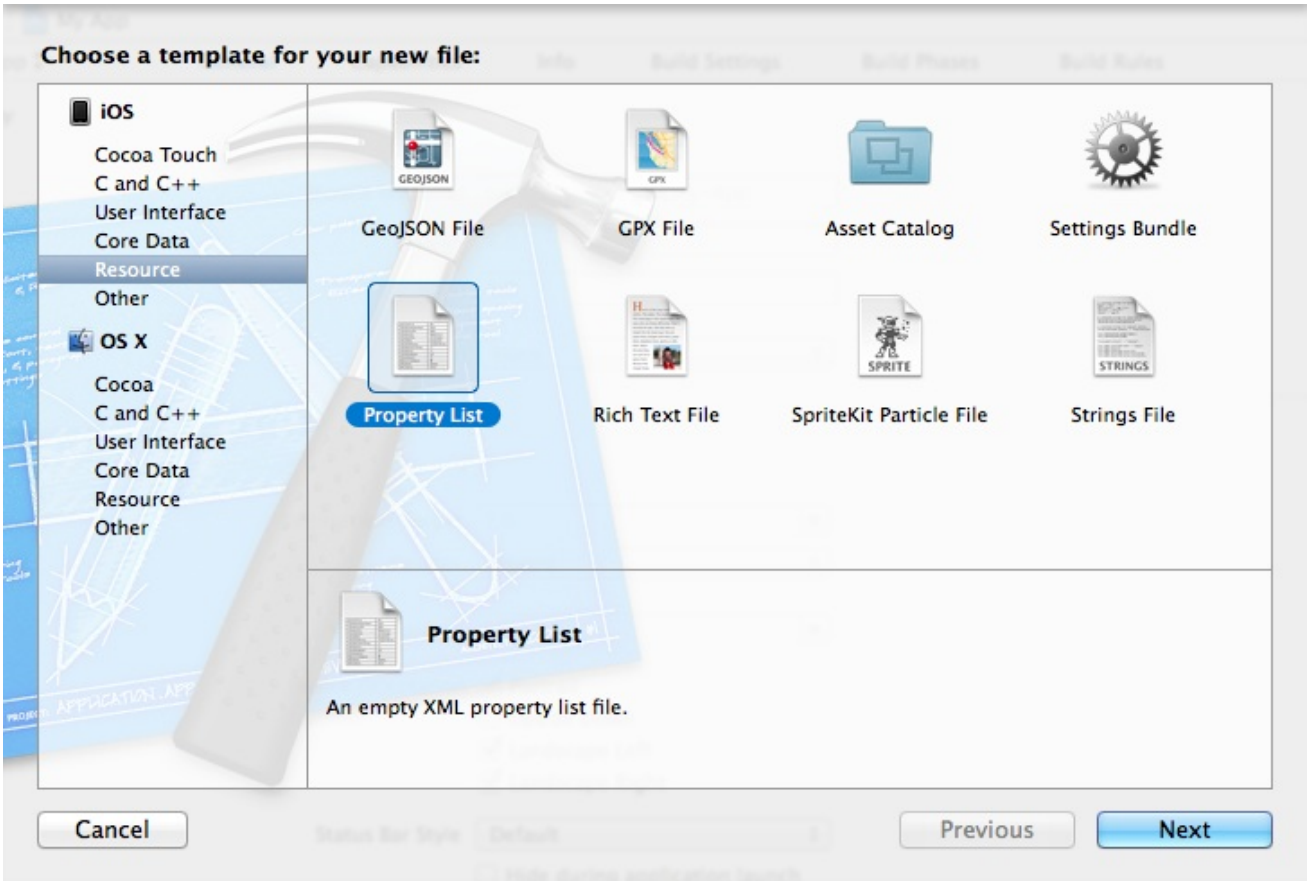
You will also need to add a Linker Flag to your app. To add this, go to the *Build Settings* screen of your project's target. Use the search to find *Other linker flags*, and add the following:

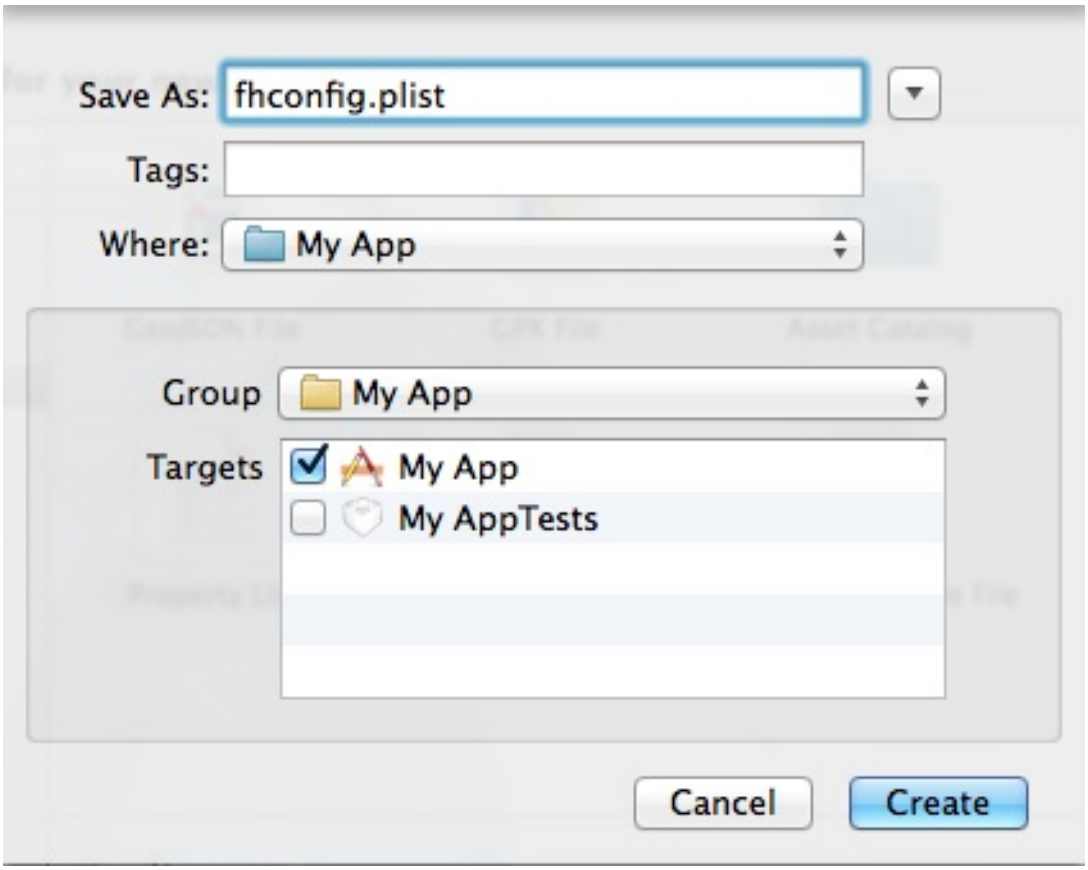
```
-Objective-C
```



Finally, add a Property List file called **`fhconfig.plist`** to your Project.


My App
2 targets, iOS SDK 7.0
Show in Finder
Open with External Editor
Open As
Show File Inspector
New File...
New Project...
Add Files to "My App"...
Delete
New Group
New Group from Selection
Sort by Name
Sort by Type
Find in Selected Groups...
Source Control
Project Navigator Help





2.2.4. Setup

You must define the properties which allow your app to communicate with RHMAP servers. Add the following values to the **`fhconfig.plist`** configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>host</key>
  <string>__RHMAP_Core_host__</string>
  <key>appid</key>
  <string>__Client_App_ID__</string>
  <key>projectid</key>
  <string>__Project_ID__</string>
  <key>appkey</key>
  <string>__Client_App_ API_key___</string>
  <key>connectiontag</key>
  <string>__Connection_tag__</string>
</dict>
</plist>
```

See Projects - Connections for more information on connections between Client Apps and Cloud Apps.

2.2.5. Initialization

Before invoking any cloud requests from the RHMAP iOS Swift SDK, you must first initialize it. Copy the following code snippet to your App Delegate's **`didFinishLaunchingWithOptions:`** method, or any other location which ensures the code is called before any cloud requests:

```
#import <FH/FH.h>
#import <FH/FHResponse.h>

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
  // Override point for customization after application launch.

  // Call a cloud side function when init finishes
  void (^success)(FHResponse *)=^(FHResponse * res) {
    // Initialisation is now complete, you can now make FHActRequest's
    NSLog(@"SDK initialised OK");
  };

  void (^failure)(id)=^(FHResponse * res){
    NSLog(@"Initialisation failed. Response = %@", res.rawResponse);
  };

  //View loaded, init the library
  [FH initWithSuccess:success AndFailure:failure];

  return YES;
}
```

CHAPTER 3. NATIVE IOS (SWIFT)

3.1. DOWNLOAD

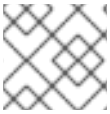
- SDK
- Sample App

WARNING

If you use an enterprise distribution certificate to sign a Swift-based iOS app built using the Build Farm, the resulting app may crash upon startup.

If you encounter this problem, refer to the Red Hat Knowledge Base article Swift-based iOS application crashes upon startup when signed using an enterprise distribution certificate without Organisational Unit field for detailed instructions on how to resolve the problem.

3.1.1. CocoaPods



NOTE

CocoaPods is available only on OSX.

CocoaPods is a Swift And Objective-C dependency manager for Xcode projects. It is built with Ruby, which comes pre-installed with OSX. It is used to distribute the RHMAP iOS Swift and iOS Objective-C Client SDKs.

3.1.1.1. Install CocoaPods Using RubyGems

To install CocoaPods using the RubyGems package manager, execute the following command. This requires the **sudo** command to be enabled in OSX.

```
sudo gem install cocoapods
```

As an optional part of the CocoaPods setup, you can store your Podspec metadata locally at **~/.cocoapods/repos**. This helps to increase the dependency resolution speed and shortens the build time for your apps. To clone the spec-repo and create the directory, execute the following command:

```
pod setup
```

3.1.1.2. Install CocoaPods Without Using sudo.

Alternatively, to install CocoaPods without using sudo, follow the *Sudo-less Installation* section of the CocoaPods Getting Started Guide.

3.1.1.3. Install the Required Plugins

The RHMAP iOS Objective-C and Swift SDK packages rely on the **`cocoapods-packager`** and **`cocapods-appledoc`** plugins. To install both plugins, execute the following command:

```
[sudo] gem install cocoapods-packager cocoapods-appledoc
```

3.1.1.4. Enable CocoaPods in an Xcode Project

To enable Cocoapods Xcode app project after installation execute the following commands:

1. To navigate to the folder of your Xcode project, use:

   ```
   cd <project_directory>
   ```

2. To create a podfile (if it does not already exist) in your project folder and automatically populate it with targets specified within the project, execute the following command:

   ```
   pod init
   ```

3.1.1.5. Install Dependencies Using CocoaPods

To install the dependencies defined in the podfile of Xcode project using CocoaPods, execute the following command:

```
pod install
```

3.2. GET STARTED

This SDK lets you use RHMAP APIs in Swift apps for iOS version 8 or higher.

The RHMAP iOS Swift SDK is an open-source project hosted in the FeedHenry iOS SDK repository on Github. Feel free to fork it and make a contribution to this project.

Before using this SDK:

- Install the Xcode IDE. You can download Xcode from the Apple Developer Portal.
- Install the CocoaPods dependency management system as described in Section 3.1.1, “CocoaPods”.

3.2.1. New App

Clone the sample app to get started with a new iOS application which has the RHMAP iOS Swift SDK included as a CocoaPods dependency.

```
git clone https://github.com/feedhenry-templates/blank-ios-swift.git
cd blank-ios-app
```

Fetch the dependencies defined in the Podfile:

```
pod install
```

Open the **`blank-ios-app.xcworkspace`** workspace in Xcode. The required dependencies are located in the Pods group.

3.2.2. Existing App

If your app does not have a Podfile already, create a new file named **`Podfile`** at the root of your project with the following contents:

```
source 'https://github.com/CocoaPods/Specs.git'

project 'ProjectName.xcodeproj'
platform :ios, '8.0'
use_frameworks!

target 'TargetName' do
    pod 'FeedHenry', '4.1.1'

end
```

Replace '4.1.1' with the version of RHMAP iOS Swift SDK you are targeting, ProjectName.xcodeproj with the name of your project and TargetName with the name of your target. If you do not specify a version number, the latest version in the CocoaPods central repository will be used.

Fetch the dependencies defined in the Podfile:

```
pod install
```

You can now open **`ProjectName.xcworkspace`** in Xcode.

3.2.3. Setup

You must define the properties which allow your app to communicate with RHMAP servers. Add the following values to the **`fhconfig.plist`** configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>host</key>
  <string>__RHMAP_Core_host__</string>
  <key>appid</key>
  <string>__Client_App_ID__</string>
  <key>projectid</key>
  <string>__Project_ID__</string>
  <key>appkey</key>
  <string>__Client_App_ API_key___</string>
  <key>connectiontag</key>
  <string>__Connection_tag__</string>
</dict>
</plist>
```

See Projects - Connections for more information on connections between Client Apps and Cloud Apps.

3.2.4. Initialization

Before invoking any cloud requests from the RHMAP iOS Swift SDK, you must first initialize it. Copy the following code snippet to your App Delegate's **`application(_, didFinishLaunchingWithOptions:)`** method, or any other location which ensures the code is called before any cloud requests:

```
import FeedHenry

func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool {
  FH.init { (resp:Response, error: NSError?) -> Void in
    if let error = error {
      print("Error: \(error)")
      return
    }
    print("Response: \(resp.parsedResponse)")
  }
  return true
}
```

3.3. API DOCUMENTATION

See the API Docs - documentation for all Client APIs

3.4. SWIFT 2.3 AND SWIFT 3

This release of RHMAP includes Swift 2.3 and Swift 3.0 functionality. Using Swift 2.3 or Swift 3.0 allows you to use Xcode 8.x and iOS 10 specific features, such as iMessage app extensions. The following templates are available:

- https://github.com/feedhenry-templates/helloworld-ios-swift/tree/swift2.3
- https://github.com/feedhenry-templates/helloworld-ios-swift/tree/swift3

3.4.1. Migrating Apps to Swift 2.3

To migrate your app to Swift 2.3:

1. Review the Swift migration guide.
2. Make sure your code is working before migrating.
3. Change the Podfile contents to the following:

```
pod 'FeedHenry', '4.2.1'
```

4. Install the pod:

```
pod install
```

5. Open the Xcode 8.x workspace by double clicking on the ProjectName.xcworkspace.
6. When prompted to migrate the code, choose Swift 2.3.

7. Build your app locally and validate that everything is working.

3.4.2. Migrating Apps to Swift 3.0

To migrate your app to Swift 3.0:

1. Review the Swift migration guide. Note the following Swift 3 coding style changes:
   - Name functions and methods according to their side-effects.
     - Those without side-effects should read as noun phrases, for example, **`x.distance(to: y)`**, **`i.successor()`**.
     - Those with side-effects should read as imperative verb phrases, for example, **`print(x)`**, **`x.sort()`**, **`x.append(y)`**.
   - Names of types and protocols are **`UpperCamelCase`**.
   - Everything else is **`lowerCamelCase`**.
2. Make sure your code is working before migrating.
3. Change the Podfile contents to the following:

```
pod 'FeedHenry', '5.0.3'
```

4. Install the pod:

```
pod install
```

5. Open the Xcode 8.x workspace by double clicking on the ProjectName.xcworkspace.
6. When prompted to migrate the code, choose Swift 3. Note the following:
   - It is unlikely that the app code will compile first time. Fix your code until it passes all unit tests.
   - Be aware of the following:
     - **`sort()`** on Array is now a mutable function
     - the difference between **`Any`** and **`AnyObject`**
     - the difference between **`Data`** and **`NSData`**
   - Consider your API label names using the Swift Api Design Guidelines.
7. Build your app locally and validate that everything is working.

CHAPTER 4. NATIVE WINDOWS

Download

- SDK
- Sample App

API Documentation

- RHMAP Client API - documentation for all RHMAP Client APIs
- RHMAP .NET SDK Reference - generated documentation of classes and methods of the RHMAP .NET SDK

4.1. INTRODUCTION

This is a standard Windows Phone Native App.

The SDK itself is an open source project that is hosted here. Feel free to fork it and make contribution to this project.

Before using this SDK, make sure you have Windows Phone developer tools installed. You can download them from here.

4.2. NUGET

NuGet is a dependency management tool for the Microsoft development platform including .NET and is used to share and distribute C# code. It serves as the primary tool for distributing the RHMAP .NET Client SDK.

4.2.1. Install the NuGet CLI using the Installer

To install NuGet on Windows-based Systems, download and run the NuGet Commandline installer.

4.2.2. Install NuGet Using Chocolatey

If you are using Chocolatey, you can install NuGet by executing the following command on the Windows command line:

```
choco install nuget.commandline
```

4.3. NEW APP

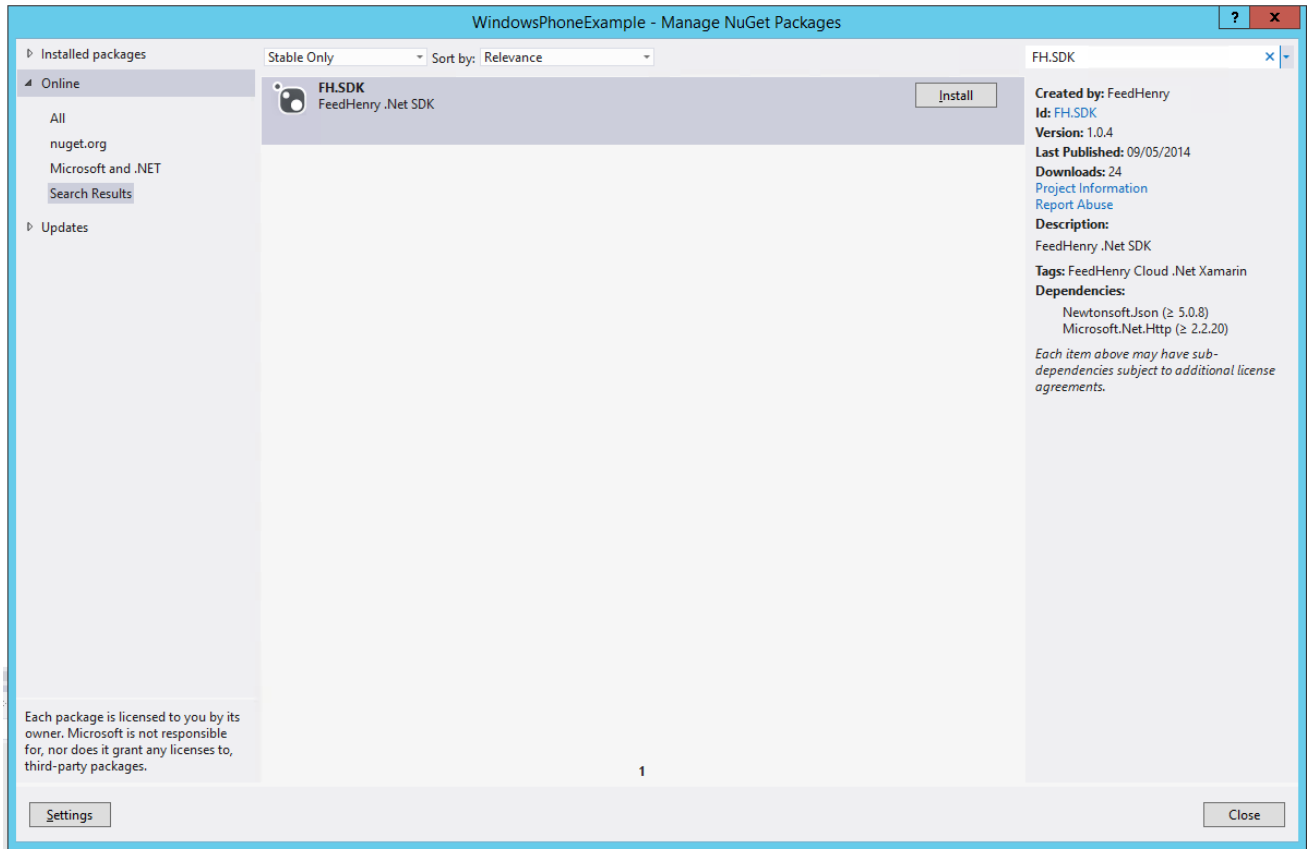
Download the sample app to get started with a new Windows Phone App which has the RHMAP SDK already included.

4.4. EXISTING APP

You can install the SDK to your project either automatically (using NuGet) or manually.

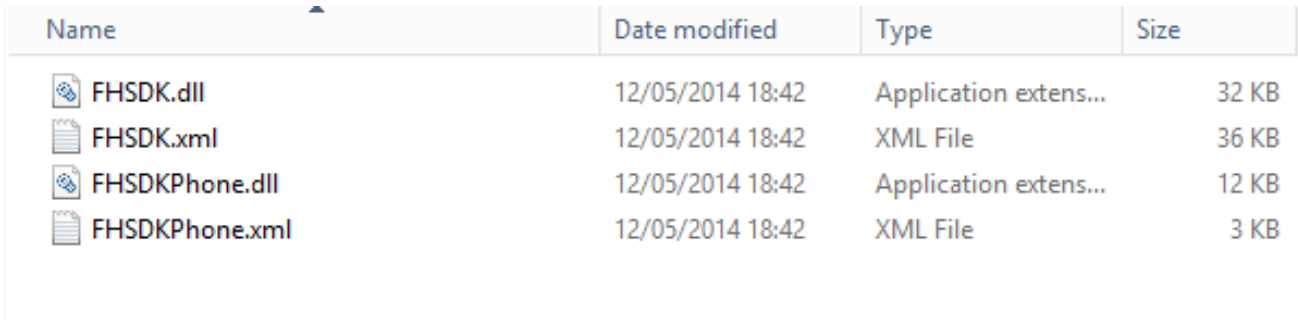
4.4.1. NuGet (Recommended)

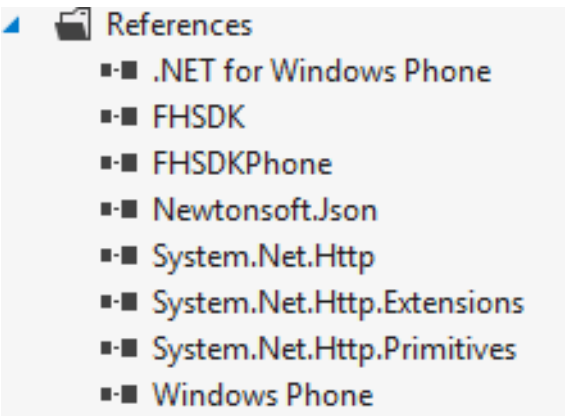
FH SDK is available on NuGet: https://www.nuget.org/packages/FH.SDK/. If you are using the NuGet plugin inside Visual Studio, search for FH.SDK. NuGet will install dependecy libraries automatically.



4.4.2. Manually

Download the SDK and unzip it. Adding the `.dll` assembly files inside the `wp80` folder as references to your project.





If you are developing a Portable Class Library project, only reference the **`FHSDK.dll`** file.

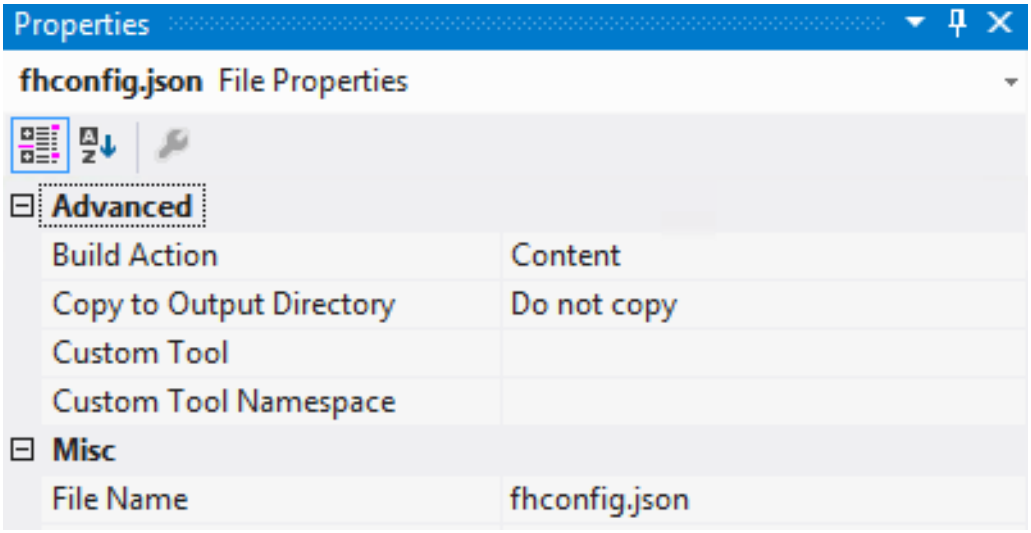
The SDK is depending on Json.Net and Microsoft HTTP Client Libraries. You need to install the assemblies of those libraries as well if they are not available in your project.

4.4.3. Set up Configuration

You need to create a new file called **`fhconfig.json`** in your project. The content of the file should be look like this:

```
{
  "appid": "__ID_OF_APP_IN_PROJECT__",
  "appkey": "__APP_API_KEY_OF_APP_IN_PROJECT__",
  "connectiontag": "__CONNECTION_TAG_TO_USE_FOR_CLOUD__",
  "host": "__APP_STUDIO_HOST__",
  "projectid": "__PROJECT_ID__"
}
```

Make sure the build action of the file is **Content**.



4.4.4. Initialise

To use the RHMAP .NET SDK, you will need to initialise the SDK like this (normally when app is started).

```
try
{
  bool inited = await FHClient.Init();
  if(inited) {
    //Initialisation is successful
  }
}
catch(FHException e)
{
  //Initialisation failed, handle exception
}
```

More information on connections can be found here.

NOTE

The **`Init`** method is the only one that is called using **`FHClient`** class, and is the only one that needs to be called from a platform-specific project (for example, can not be called from a PCL project).

All the other SDK methods are called using **`FH`** class which is defined in the **`FHSDK.dll`** assembly. This assembly can be referenced by other PCL projects. This way if your cross-platform solution contains a PCL project, you can reference this assembly file and call SDK functions from there.

4.5. USE YOUR OWN CHOICE OF HTTPCLIENT

By default, the .NET SDK will use the Microsoft HTTP Client Libraries to perform all the HTTP requests. However, if you are developing iOS and Android apps using Xamarin, the ModernHttpClient is a better choice. If you want to use that, all you have to do is to install the **`ModernHttpClient`** component in your app, then use it like this:

```
//the following should be called BEFORE FHClient.Init is called
//use ModernHttpClient on Android
FHHttpClientFactory.Get = (() => new HttpClient(new
OkHttpNetworkHandler()));
```

If you don't like either of these, you can use whatever HTTP (or REST) client you like. All you need is the cloud host of the app, which you can get using the following method:

```
string cloudHost = FH.GetCloudHost();
```

However, the downside of the approach is that your app won't be able to use the analytics service provided by the platform as some meta data is missing in the requests. To re-enable that, all you have to do is to add the meta data returned by the following method as a set of headers to each HTTP request:

```
IDictionary<string, string> metaData = FH.GetDefaultParamsAsHeaders();
HttpRequestMessage requestMessage = new HttpRequestMessage(...);
//then loop through the metaData and add each entry as a http header to
your request, using the key as the header name and value as the header
value
foreach(var item in metaData){
  requestMessage.Headers.Add(item.Key, item.Value);
}
...
```

4.6. USE SDK

See API Docs for full details on the APIs available within the SDK.

CHAPTER 5. XAMARIN

5.1. DOWNLOAD

- SDK
- Sample App

5.2. INTRODUCTION

This is a standard Xamarin App which allows you to create Native iOS and Android apps in C#.

The SDK itself is an open source project that is hosted here. Feel free to fork it and make contribution to this project.

Before using this SDK, make sure you have Xamarin developer tools installed. You can download them from here.

To develop Windows Phone app, you need to have Windows Phone Developer tools installed. You can download it from here.

We recommend you install Xamarin For Visual Studio as well.

5.3. NUGET

NuGet is a dependency management tool for the Microsoft development platform including .NET and is used to share and distribute C# code. It serves as the primary tool for distributing the RHMAP .NET Client SDK.

5.3.1. Install the NuGet CLI using the Installer

To install NuGet on Windows-based Systems, download and run the NuGet Commandline installer.

5.3.2. Install NuGet Using Chocolatey

If you are using Chocolatey, you can install NuGet by executing the following command on the Windows command line:

```
choco install nuget.commandline
```

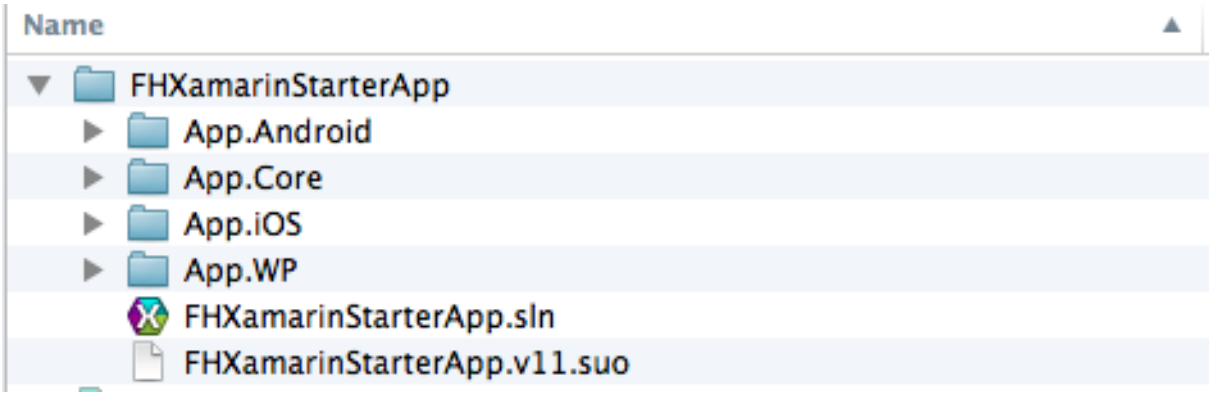
5.4. NEW APP

Download the sample app to get started with a new Xamarin App which has the RHMAP SDK already included.

The app contains 4 sub-projects. It is setup to use Portable Class Libraries to share code across all the apps. More details about this approach can be found here.

- **`App.Core`** - A PCL project. The code in this project is shared by other apps. Most of the app's business logic should be defined here.

- **`App.WP`** - A Windows Phone app project depends on the **`App.Core`** project. Normally it should contain UI code and WP-specific code.
- **`App.Android`** - An Android app project depends on the **`App.Core`** project. Normally it should contain UI code and Android-specific code.
- **`App.iOS`** - An iOS app project depends on the **`App.Core`** project. Normally it should contain UI code and iOS-specific code.

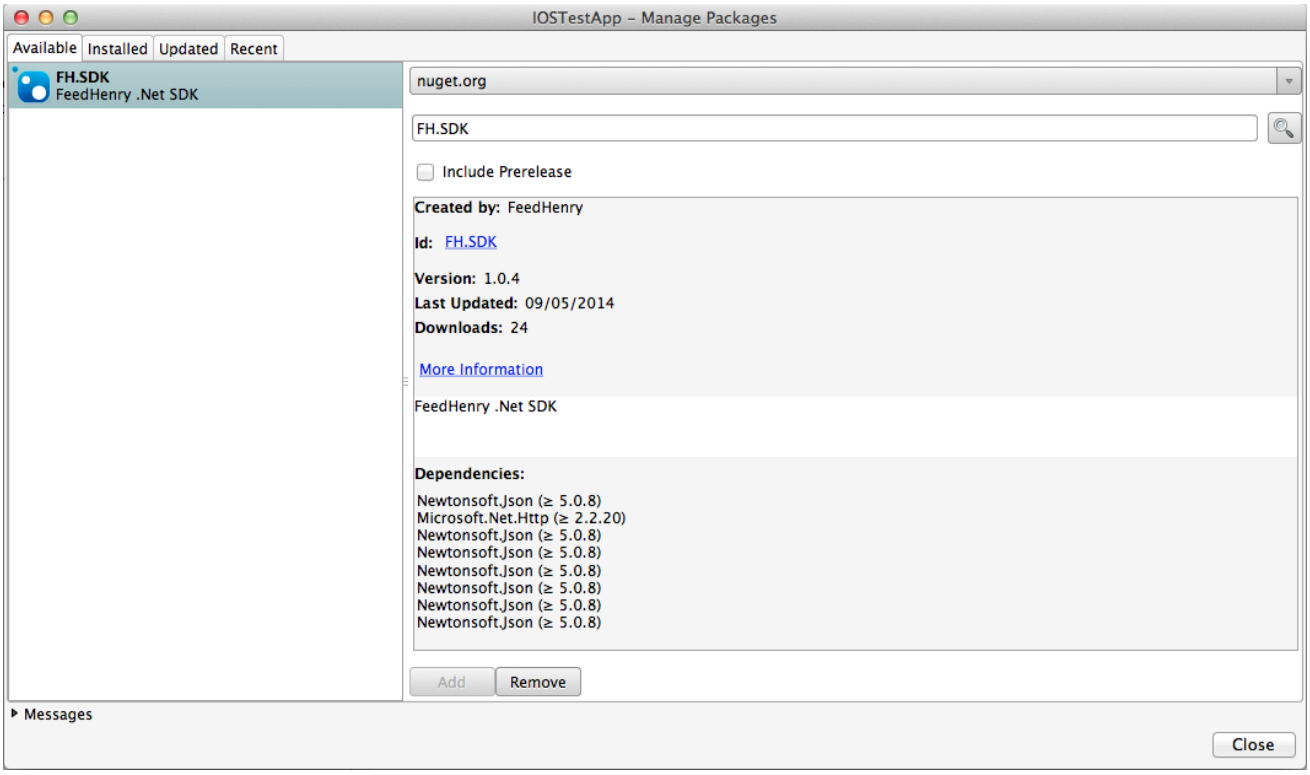


5.5. EXISTING APP

You can install the SDK to your project either automatically (using NuGet) or manually.

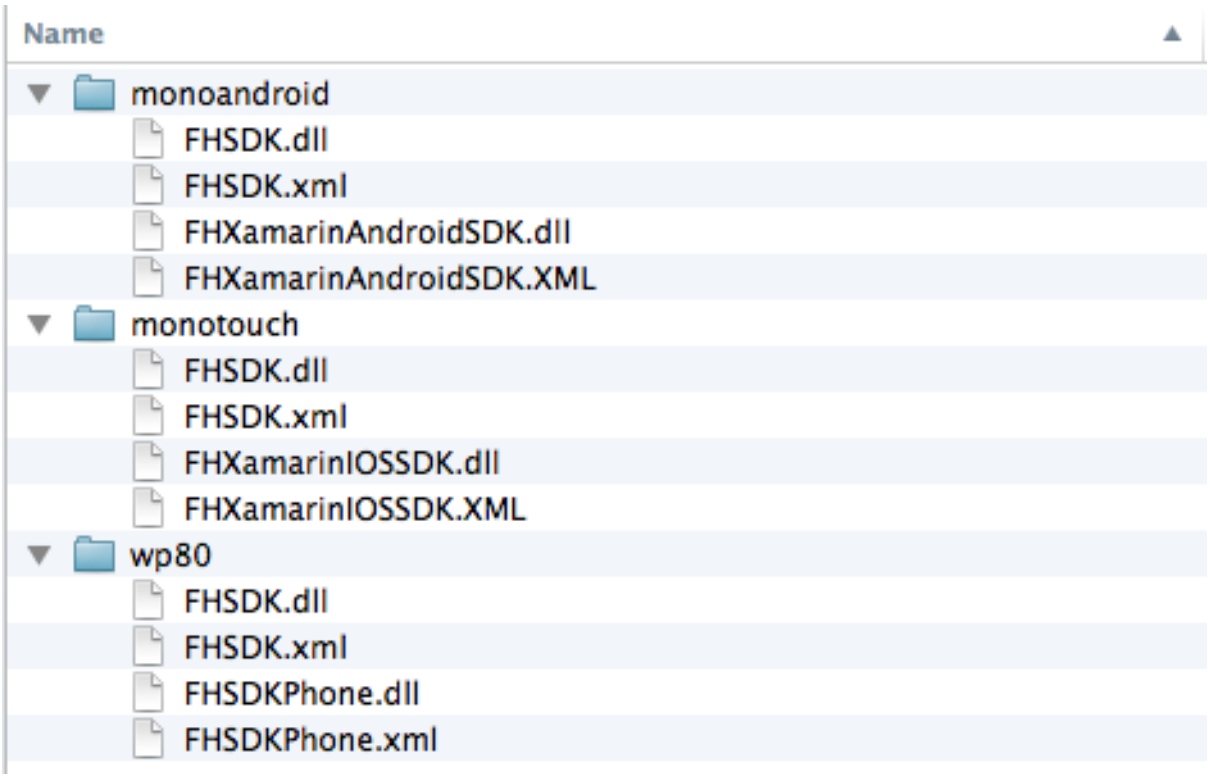
5.5.1. NuGet (Recommended)

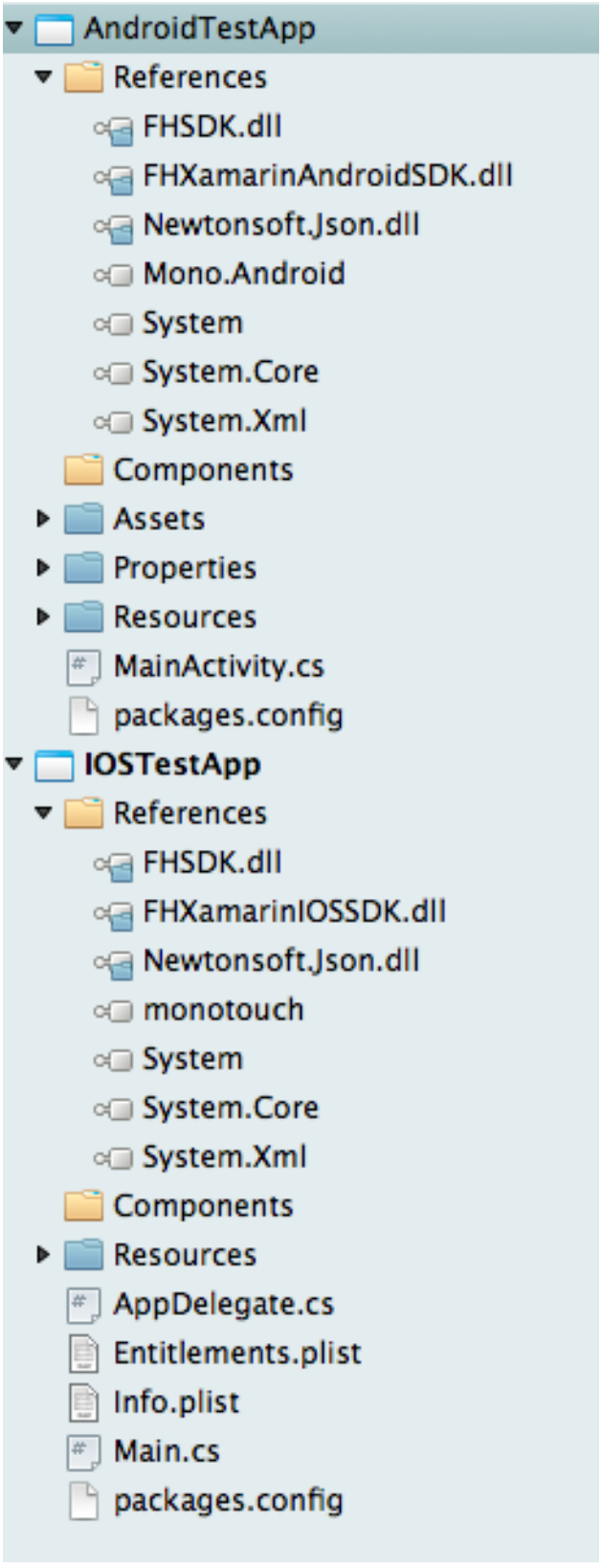
FH SDK is available on NuGet: https://www.nuget.org/packages/FH.SDK/. If you are using the NuGet plugin inside Xamarin Studio, search for FH.SDK. NuGet will install dependecy libraries automatically.



5.5.2. Manually

Download the SDK and unzip it. Adding the `.dll` assembly files from the folder that is corresponding to your project's build target as references.





If you are developing a Portable Class Library project, only reference the **`FHSDK.dll`** file.

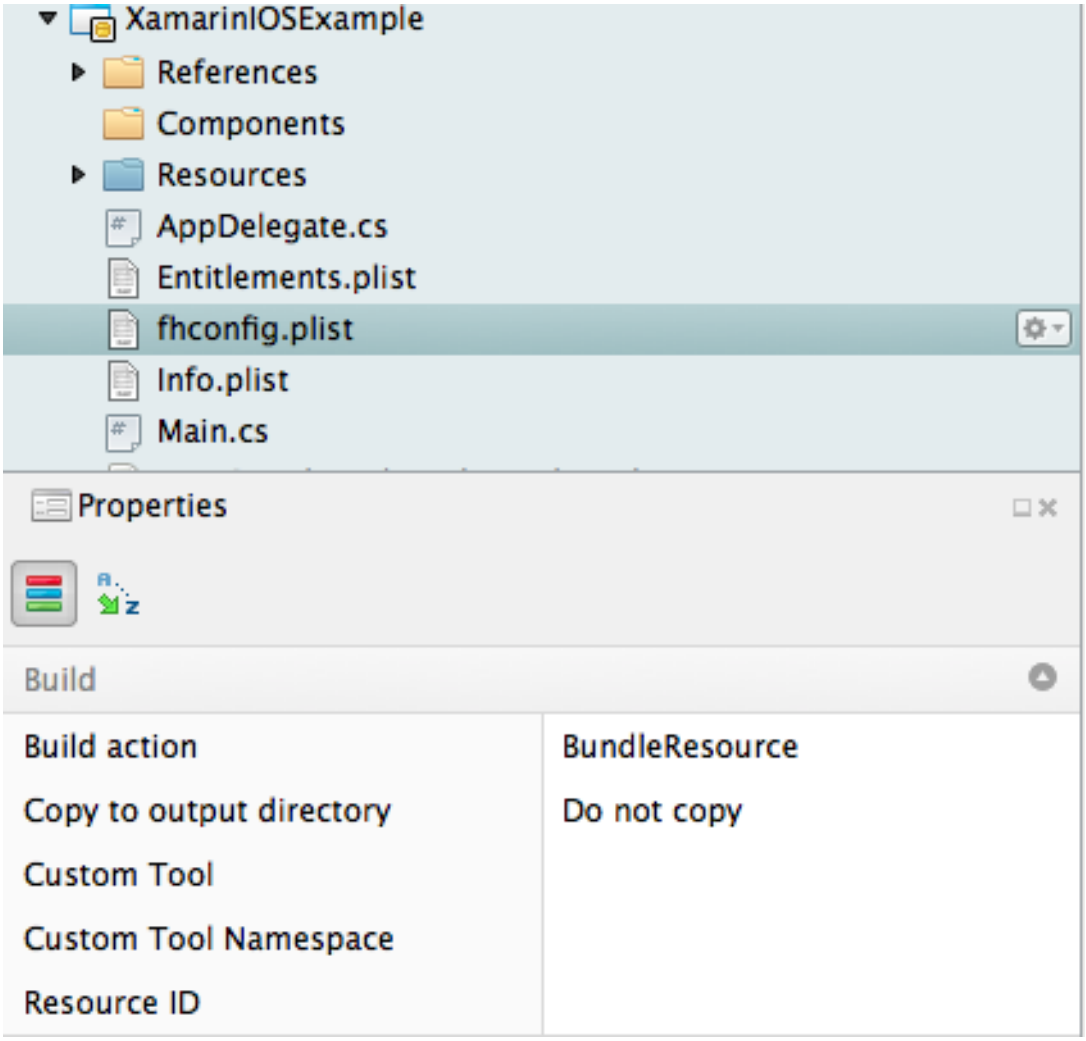
The SDK depends on Json.Net and Microsoft HTTP Client Libraries. You need to install the assemblies of those libraries as well if they are not available in your project.

5.5.3. Set up Configuration

For each platform-specific application, you need to create the corresponding configuration files. The content of each file should be the same as described in each platform's native SDK doc.

5.5.4. iOS

`fhconfig.plist` in the root of the application. Set build action to **BundleResource**.

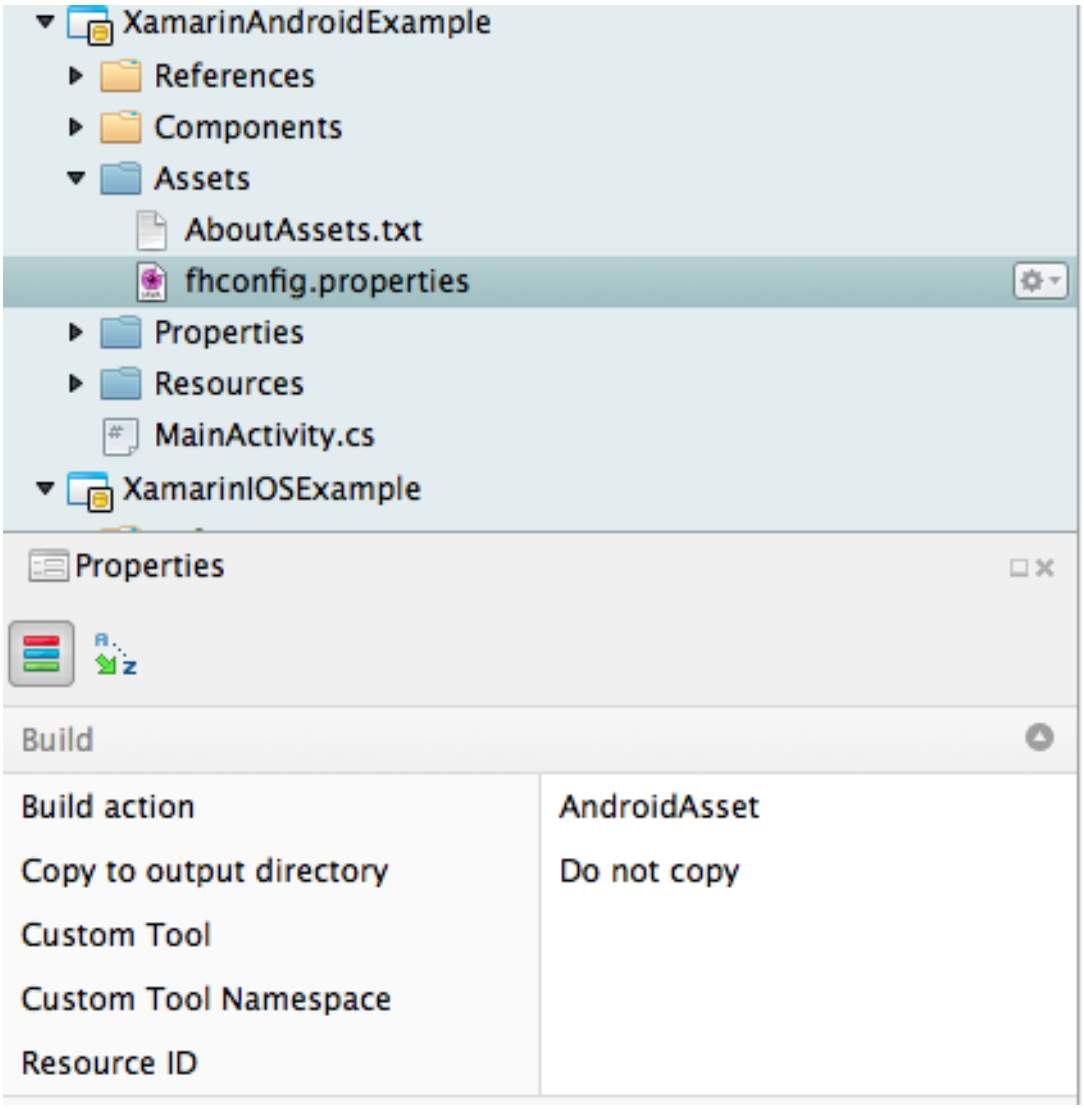


`fhconfig.plist` file contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>host</key>
  <string>__APP_STUDIO_HOST__</string>
  <key>appid</key>
  <string>__ID_OF_APP_IN_PROJECT__</string>
  <key>projectid</key>
  <string>__PROJECT_ID__</string>
  <key>appkey</key>
  <string>__APP_API_KEY_OF_APP_IN_PROJECT__</string>
  <key>connectiontag</key>
  <string>__CONNECTION_TAG_TO_USE_FOR_CLOUD__</string>
</dict>
</plist>
```

5.5.5. Android

`fhconfig.properties` in the *Assets* directory of the application. Set build action to **AndroidAsset**.

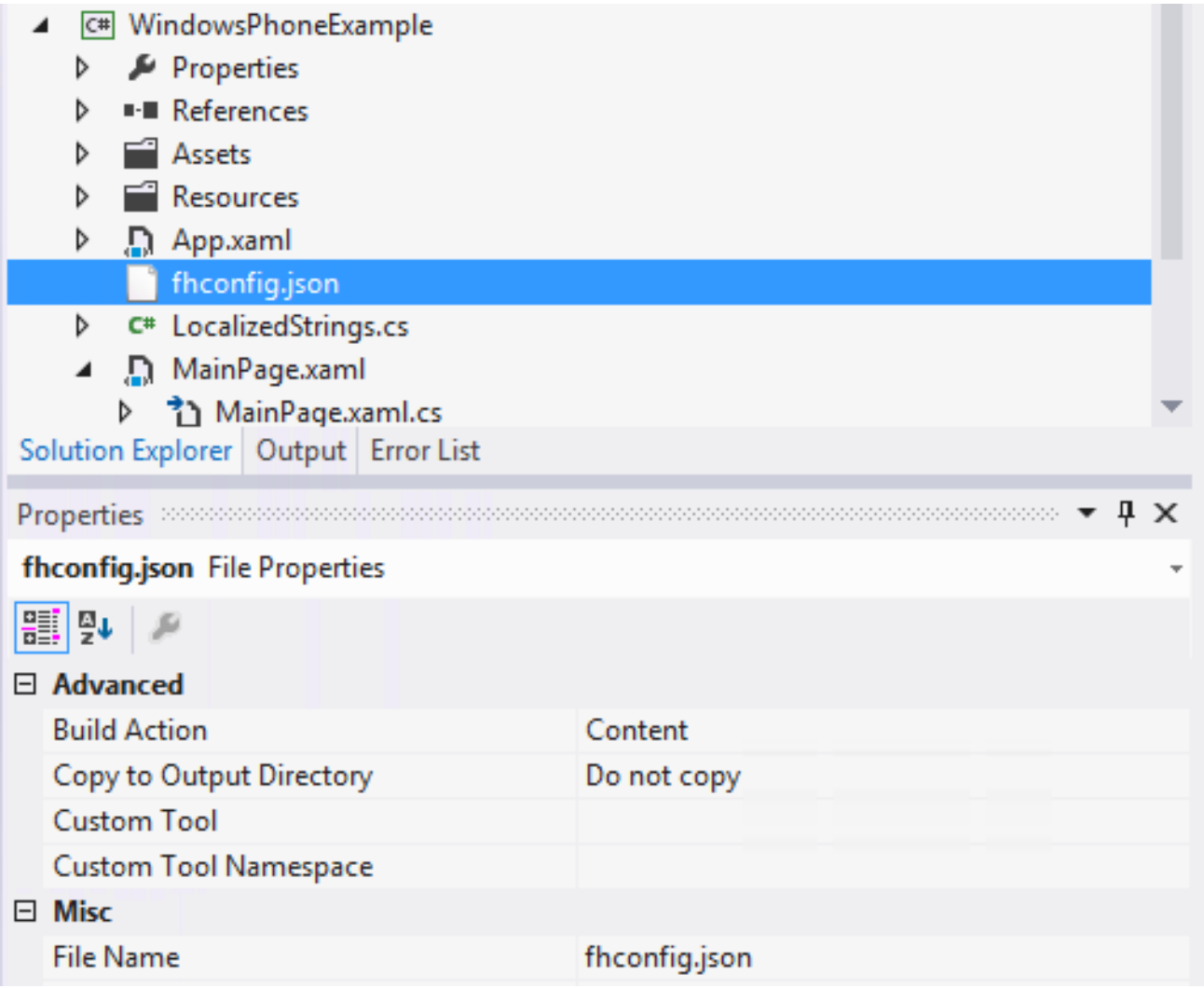


`fhconfig.properties` file contents:

```
host = __APP_STUDIO_HOST__
projectid = __PROJECT_ID__
connectiontag = __CONNECTION_TAG_TO_USE_FOR_CLOUD__
appid = __ID_OF_APP_IN_PROJECT__
appkey = __APP_API_KEY_OF_APP_IN_PROJECT__
```

5.5.6. Windows Phone 8

`fhconfig.json` in the root of the application. Set build action to **Content**.



`fhconfig.json` file contents:

```
{
  "appid": "__ID_OF_APP_IN_PROJECT__",
  "appkey": "__APP_API_KEY_OF_APP_IN_PROJECT__",
  "connectiontag": "__CONNECTION_TAG_TO_USE_FOR_CLOUD__",
  "host": "__APP_STUDIO_HOST__",
  "projectid": "__PROJECT_ID__"
}
```

More information on connections can be found here.

5.5.7. Initialise

To use the RHMAP .NET SDK, you will need to initialise the SDK like this in the platform-specific project (not the PCL project) when app finish starting.

```
try
{
  bool inited = await FHClient.Init();
  if(inited) {
    //Initialisation is successful
  }
}
catch(FHException e)
```

```
{
  //Initialisation failed, handle exception
}
```

FHClient is available in the following namespaces:

- **`FHSDK.Phone`** — For WP8
- **`FHSDK.Droid`** — For Android
- **`FHSDK.Touch`** — For iOS

Depending on your app's build target, only one of these name spaces should be available to your app.

The main reason for having the same **`FHClient`** class defined in different name spaces is to ensure that the platform-specific assembly file is loaded correctly.

NOTE

The **`Init`** method is the only one that is called using **`FHClient`** class, and is the only one that needs to be called from a platform-specific project (for example, Can not be called from a PCL project).

All the other SDK methods are called using **`FH`** class which is defined in the **`FHSDK.dll`** assembly. This assembly can be references by other PCL projects. This way if your cross-platform solution contains a PCL project, you can reference this assembly file and call SDK functions from there.

5.6. USE YOUR OWN CHOICE OF HTTPCLIENT

By default, the .NET SDK will use the Microsoft HTTP Client Libraries to perform all the http requests. However, if you are developing iOS and Android apps using Xamarin, the ModernHttpClient is a better choice. If you want to use that, all you have to do is to install the ModernHttpClient component in your app, then use it like this:

```
//the following should be called BEFORE FHClient.Init is called
//use ModernHttpClient on Android
FHHttpClientFactory.Get = (() => new HttpClient(new
OkHttpNetworkHandler()));
```

If you don't like either of these, you can use whatever HTTP (or REST) client you like. All you need is the cloud host of the app, which you can get using the following method:

```
string cloudHost = FH.GetCloudHost();
```

However, the downside of the approach is that your app won't be able to use the analytics service provided by the platform as some meta data is missing in the requests. To re-enable that, all you have to do is to add the meta data returned by the following method as a set of headers to each HTTP request:

```
IDictionary<string, string> metaData = FH.GetDefaultParamsAsHeaders();
HttpRequestMessage requestMessage = new HttpRequestMessage(...);
//then loop through the metaData and add each entry as a http header to
your request, using the key as the header name and value as the header
value
```

```
foreach(var item in metaData){
  requestMessage.Headers.Add(item.Key, item.Value);
}
...
```

5.7. USE SDK

See API Docs for full details on the APIs available within the SDK.

CHAPTER 6. CORDOVA

6.1. NPM

npm is the package manager for distributing JavaScript code. It is used to install the RHMAP JavaScript SDK for developing Cordova apps.

6.1.1. Installing Node.js and npm

You must have Node.js installed on your system to use npm. It is recommended that you install NodeJS version 4.x or later, which includes npm version 2.x by default.

6.1.1.1. Install Node.js Using a Package Manager

To install Node.js using your system package manager, follow the instructions appropriate for your system on the Node.js site

To check the version of npm you are currently using, execute the following command:

```
sudo npm -v
```

6.1.2. Installing Node.js and npm on Red Hat Enterprise Linux

To install NodeJS on systems running RHEL, you must obtain the installation files from the Red Hat Software Collections. Installing NodeJS requires a subscription to the Red Hat Subscription Manager.

To install NodeJS on RHEL, follow the guide on the Red Hat Developers Portal.

6.2. GET STARTED

The RHMAP JavaScript SDK lets you use RHMAP APIs in Cordova apps, which are developed primarily using web technology — HTML, JS, CSS — while still allowing access to many device capabilities.

The underlying native project and Cordova libraries are exposed to the developer. This allows for full customisation of the application, including Cordova Plugins and third-party libraries. Typically, the amount of time spent writing or modifying native code for Cordova apps is relatively small & requires only a small subset of a development team to have experience with native code. This small subset can handle the native code and expose any additional plugins or SDKs through a JavaScript API using the Cordova plugin architecture.

See the official Cordova website for more information on Cordova.

6.2.1. New App

Download the sample app to get started with a new Cordova App which has the RHMAP JavaScript SDK already included.

6.2.2. Existing App

NOTE

`fh-js-sdk` is distributed using npm. To be able to complete the steps below, you must install npm.

1. In the root folder of your app, open the **`package.json`** file and add the latest version of **`fh-js-sdk`** onto the list of dependencies:

```
"dependencies": {
  "fh-js-sdk": "^2.17.5",
}
```

2. In the root folder of your app, execute the following command to install the SDK using npm:

```
sudo npm install fh-js-sdk
```

   This installs the SDK version specified in the dependencies in the **`package.json`** file

3. Add the following code to your **`index.html`** file:

```
<head>
  <script src="feedhenry.js" type="text/javascript"></script>
</head>
```

6.2.3. Setup

Finally, you need to define properties which allow your app to communicate with RHMAP servers. Create a new file in the same directory as the SDK file, called **`fhconfig.json`**, with the following contents, replacing the references in brackets with values from your project:

```
{
  "host": "<RHMAP Core host>",
  "projectid": "<Project ID>",
  "connectiontag": "<Connection tag>",
  "appid": "<Client App ID>",
  "appkey": "<Client App API key>"
}
```

See Projects - Connections for more information on connections between Client Apps and Cloud Apps.

6.3. COMPATIBILITY

To ensure a working setup for developing Cordova apps, we advise that your local Cordova CLI version matches the version of Cordova installed in the RHMAP Build Farm and that your Cordova platform versions are within the supported range. The Supported Configurations page lists the Cordova CLI and platform versions supported by the Build Farm.

If you don't specify Cordova platform versions in your app, the default pinned versions for the Cordova CLI used in Build Farm will be used. We advise you to always specify platform versions in your Cordova apps to ensure build reproducibility and stability.

6.4. API DOCUMENTATION

- API Docs - documentation for all Client APIs
- JS SDK - JavaScript SDK

CHAPTER 7. WEB

7.1. NPM

npm is the package manager for distributing JavaScript code. It is used to install the RHMAP JavaScript SDK for developing Cordova apps.

7.1.1. Installing Node.js and npm

You must have Node.js installed on your system to use npm. It is recommended that you install NodeJS version 4.x or later, which includes npm version 2.x by default.

7.1.1.1. Install Node.js Using a Package Manager

To install Node.js using your system package manager, follow the instructions appropriate for your system on the Node.js site

To check the version of npm you are currently using, execute the following command:

```
sudo npm -v
```

7.1.2. Installing Node.js and npm on Red Hat Enterprise Linux

To install NodeJS on systems running RHEL, you must obtain the installation files from the Red Hat Software Collections. Installing NodeJS requires a subscription to the Red Hat Subscription Manager.

To install NodeJS on RHEL, follow the guide on the Red Hat Developers Portal.

7.2. INTRODUCTION

This is a Node.js + Express web application. These apps provide more advanced desktop/tablet web portals and mobile websites.

They expose the full power of Node.js for web app development including functionality such as Express 4 and server side templating using template engines such as ejs.

They also support static file serving for standard HTML5, CSS and JavaScript.

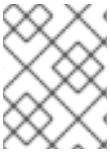
7.3. NEW APP

Download the sample app to get started with a new Web App which has the RHMAP SDK already included.

7.4. EXISTING APP

Web Apps apps use our standard JavaScript SDK — included in the **`index.html`** file.

7.4.1. Download SDK



NOTE

`fh-js-sdk` is distributed using npm. To be able to complete the steps below, you must install npm.

1. In the **`/public`** folder of your app project, open the **`package.json`** file and add the latest version of **`fh-js-sdk`** onto the list of dependencies:

```
"dependencies": {
  "fh-js-sdk": "^2.17.5",
}
```

2. In the root folder of your app, execute the following command to install the SDK:

```
sudo npm install fh-js-sdk
```

This installs the SDK version specified in the dependencies in the **`package.json`** file

7.4.2. Integrate SDK

Add the following code to your **`index.html`** file.

```
<head>
  <script src="feedhenry.js" type="text/javascript"></script>
</head>
```

7.4.3. Set up Configuration

Create a new file in the same directory as the SDK file (**`feedhenry.js`**) called **`fhconfig.json`**, with the following contents:

```
{
  "appid": "__ID_OF_APP_IN_PROJECT__",
  "appkey": "__APP_API_KEY_OF_APP_IN_PROJECT__",
  "connectiontag": "__CONNECTION_TAG_TO_USE_FOR_CLOUD__",
  "host": "__APP_STUDIO_HOST__",
  "projectid": "__PROJECT_ID__"
}
```

More information on connections can be found here.

7.5. USE SDK

See API Docs for full details on the APIs available within the SDK.