



Red Hat Mobile Application Platform

ホスト型 3

クライアント SDK

Red Hat Mobile Application Platform ホスト型 3 向け

Red Hat Customer Content
Services

法律上の通知

Copyright © 2017 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書では、サポートされているすべてのプラットフォーム用の RHMAP クライアント SDK を使用する方法についての情報を提供します。

目次

第1章 ネイティブ ANDROID	4
1.1. ダウンロード	4
1.2. スタート	4
1.3. 互換性	5
1.4. API ドキュメント	5
第2章 ネイティブ IOS (OBJECTIVE-C)	6
2.1. ダウンロード	6
2.2. スタート	6
2.3. API ドキュメント	13
第3章 ネイティブ IOS (SWIFT)	14
3.1. ダウンロード	14
3.2. スタート	14
3.3. API ドキュメント	16
第4章 ネイティブ WINDOWS	17
4.1. ダウンロード	17
4.2. はじめに	17
4.3. 新規アプリ	17
4.4. 既存アプリ	17
4.5. HTTPCLIENT の独自オプションの使用	20
4.6. SDK の使用	20
第5章 XAMARIN	21
5.1. ダウンロード	21
5.2. はじめに	21
5.3. 新規アプリ	21
5.4. 既存アプリ	22
5.5. HTTPCLIENT の独自オプションの使用	28
5.6. SDK の使用	29
第6章 CORDOVA	30
6.1. ダウンロード	30
6.2. スタート	30
6.3. 互換性	31
6.4. API ドキュメント	31
第7章 CORDOVA LIGHT	32
7.1. ダウンロード	32
7.2. スタート	32
7.3. API ドキュメント	33
第8章 WEB	34
8.1. ダウンロード	34
8.2. はじめに	34
8.3. 新規アプリ	34
8.4. 既存アプリ	34
8.5. SDK の使用	35
第9章 APPCELERATOR	36
9.1. ダウンロード	36
9.2. はじめに	36
9.3. 新規アプリ	36

9.4. 既存アプリ	36
9.5. SDK の使用	37

第1章 ネイティブ ANDROID

1.1. ダウンロード

🔗 [SDK](#)

🔗 [サンプルアプリ](#)

1.2. スタート

この SDKにより、Android M (バージョン 6.0、API level 23) までの Android アプリで RHMAP API を使用することができます。

RHMAP Android SDK は、Github の [FeedHenry Android SDK リポジトリ](#) でホストされるオープンソースプロジェクトです。ご自由にプロジェクトのフォークを行い、このプロジェクトに貢献してください。

この SDK を使用する前に、Android SDK または Android Studio がインストールされていることを確認してください。これらは、公式の [Android 開発者ポータル](#) からダウンロードできます。

1.2.1. 新規アプリ

[サンプルアプリ](#) をダウンロードし、すでに RHMAP SDK が組み込まれている新規のネイティブ Android アプリケーションを使って開始します。

1.2.2. 既存アプリ

build.gradle ファイルで以下の依存関係を宣言します。

```
compile 'com.feedhenry:fh-android-sdk:3.0.0'
```

Gradle を使用していない場合は、JAR の SDK をその依存関係と共に手動でダウンロードできます。詳細は、SDK の Github リポジトリの README ファイルにある「[Usage](#)」セクションを参照してください。

1.2.3. セットアップ

SDK には、**INTERNET** パーミッションが必要です。パーミッションがまだ追加されていない場合は、アプリの **AndroidManifest.xml** に追加してください。

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

最後に、アプリが RHMAP サーバーと通信できるようにするプロパティを定義する必要があります。**assets** フォルダーに以下の内容の **fhconfig.properties** という新規ファイルを作成します。括弧内の参照部分はプロジェクトの値に置き換えます。

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


クライアントアプリとクラウドアプリ間の通信についての詳細は、「[Projects - Connections](#)」を参照してください。

1.2.4. 初期化

RHMAP SDK からのクラウド要求を呼び出す前に、まず、以下のように初期化する必要があります。

```
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());
    }
});
```

1.3. 互換性

アプリが Android M (API Level 23) をターゲットにする場合、バージョン 3.x 以上 (可能であれば最新バージョン) の RHMAP Android SDK を使用する必要があります。SDK の 3.x バージョンは、Apache HTTP クライアント (**org.apache.http** パッケージ) が Android M の時点で Android SDK と共に配布されなくなるという Android SDK の大幅な変更を主な理由に導入されました。RHMAP SDK は、HTTP クライアントとして代わりに **cz.msebera.android:httpclient** を使用しています。

既存のアプリをターゲットの Android M にアップグレードする場合、新規 HTTP クライアントを使用できるようにアプリをアップグレードする必要もあります。Gradle を使用していない場合、RHMAP SDK と共に配布され、かつ [リポジトリの deps フォルダ](#) から利用できる HTTP クライアントの JAR ファイルを使用できます。

1.4. API ドキュメント

📖 [API Docs](#): すべてのクライアント API についてのドキュメント

📖 [Javadoc](#): Android RHMAP SDK のクラスおよびメソッドについての生成ドキュメント

第2章 ネイティブ iOS (OBJECTIVE-C)

2.1. ダウンロード

※ [SDK](#)

※ [サンプルアプリ](#)

2.2. スタート

この SDK により、RHMAP でサポートされるすべての iOS バージョン用の Objective-C アプリで RHMAP API を使用することができます。

RHMAP iOS Objective-C SDK は、Github の [FeedHenry iOS SDK リポジトリ](#) でホストされるオープンソースプロジェクトです。ご自由にプロジェクトのフォークを行い、このプロジェクトに貢献してください。

この SDK を使用する前に、Xcode IDE がインストールされていることを確認してください。Xcode は [Apple 開発者ポータル](#) からダウンロードできます。

プロジェクトの依存関係を管理するために [CocoaPods](#) を使用する予定であれば、最初にこれをインストールする必要があります。

```
sudo gem install cocoapods
```

2.2.1. 新規アプリ

新規テンプレートアプリを使って開始する場合、選択肢として以下の 2 つのオプションがあります。

※ CocoaPods の使用 (推奨される方法で Studio のデフォルト)

※ FH フレームワークの使用

2.2.1.1. CocoaPods の使用

[サンプルアプリ](#) をクローンし、RHMAP SDK を CocoaPods の依存関係として組み込んだ新規 iOS アプリケーションを使って開始します。

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

[Podfile](#) で定義された依存関係を取得します。

```
pod install
```

Xcode で **blank-ios-app.xcworkspace** ワークスペースを開きます。必要な依存関係は Pods グループにあります。

2.2.1.2. FH フレームワークの使用

[サンプルアプリ](#)をクローンし、RMAP SDK がすでに組み込まれている新規 iOS アプリケーションを使って開始します。

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

Xcode で **blank-ios-app.xcworkspace** ワークスペースを開きます。

2.2.2. 既存アプリ

既存アプリと RMAP を統合する場合、選択肢として以下の 2 つのオプションがあります。

- ※ アプリが CocoaPods を使用する場合、FH SDK を新規 pod として追加する。
- ※ (そうでない場合) FH フレームワークを Xcode プロジェクトに追加する。

2.2.2.1. CocoaPods の使用

Podfile を開き、以下の依存関係を追加します。

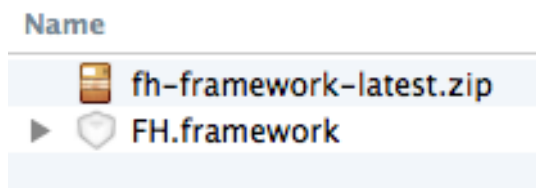
```
pod FH, '3.1.1'
```

3.1.1 を、ターゲットにしている RMAP iOS Objective-C SDK のバージョンに置き換えます。バージョン番号を指定しない場合、CocoaPods 中央リポジトリの最新バージョンが使用されます。

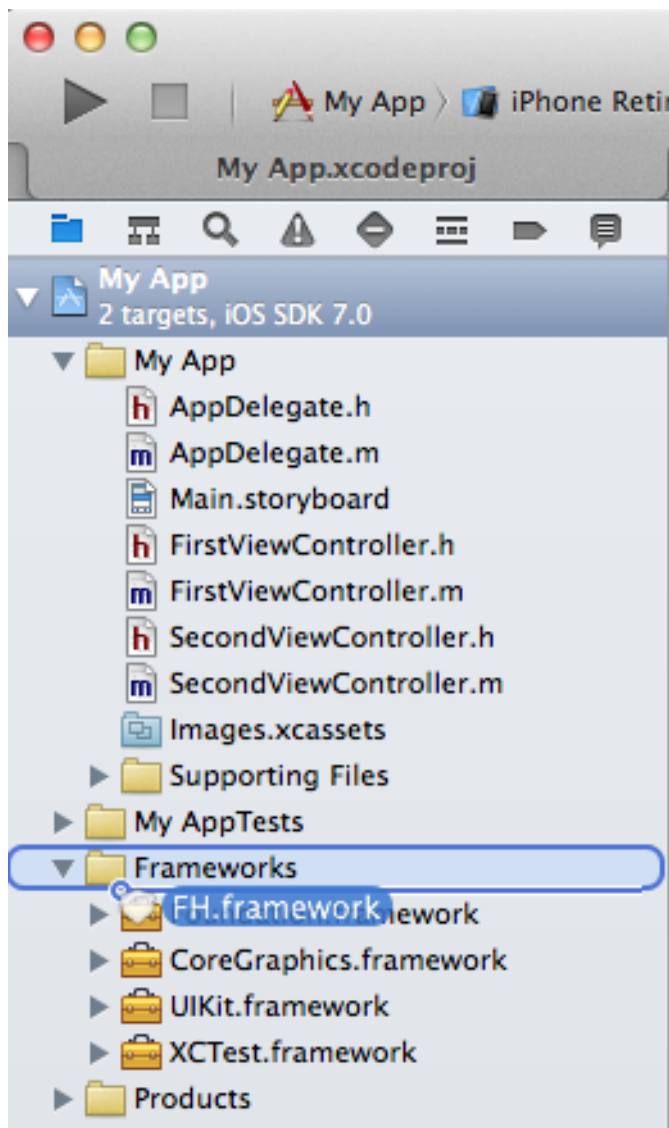
2.2.2.2. FH フレームワークの使用

SDK をダウンロードし、**fh-framework-latest.zip** として保存します。

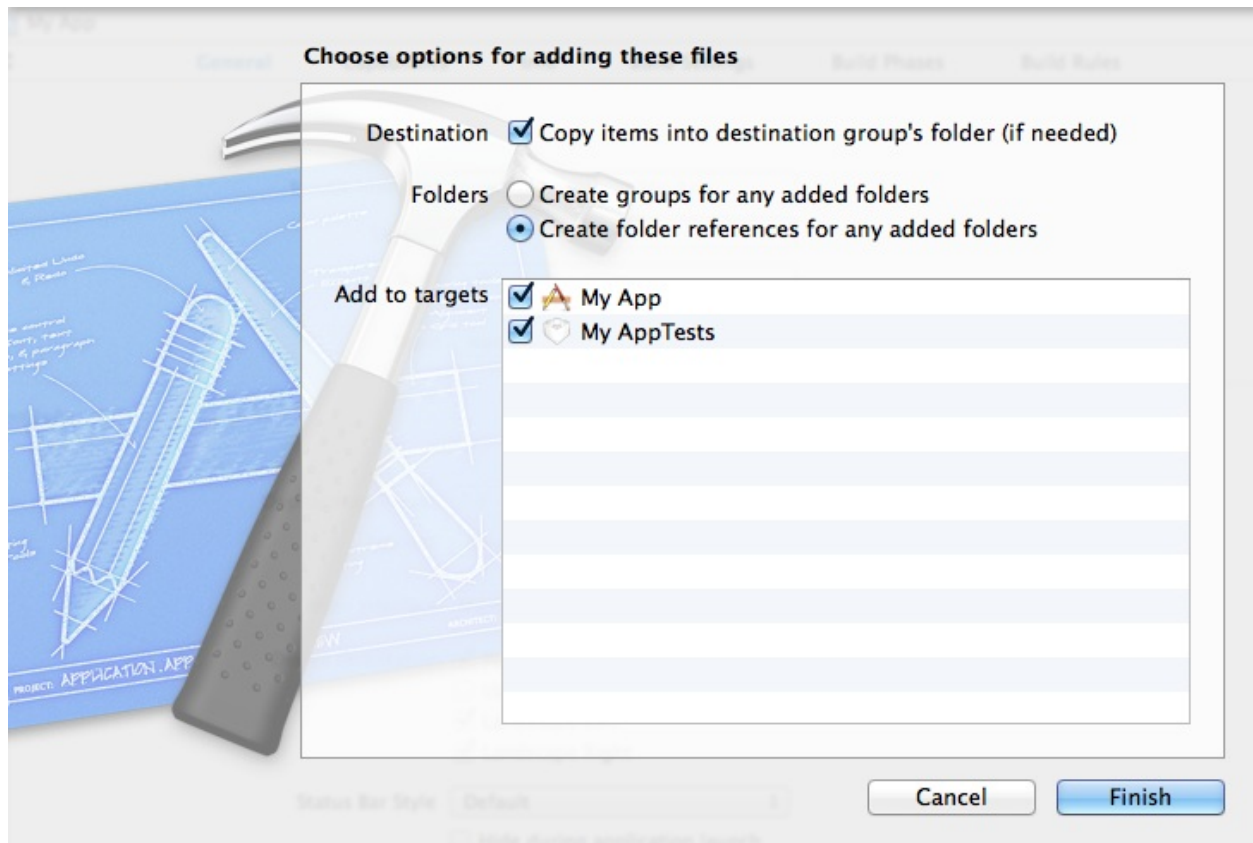
SDK を統合するには、上記のダウンロードした Zip ファイルから **FH.framework** ファイルを抽出します。



ダウンロード後は、**FH.framework** ファイルを Xcode プロジェクトにドラッグします。



ダイアログで、デフォルトを受け入れ、**Finish** をクリックします。

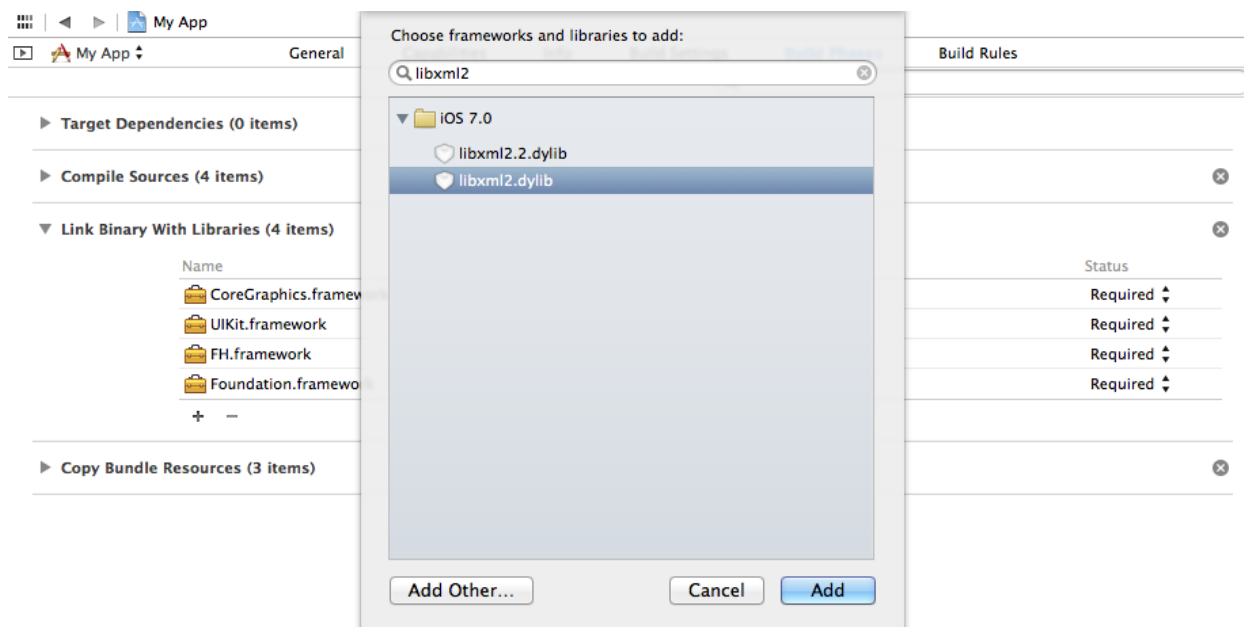


2.2.3. 統合

RHMAP iOS SDK にはフレームワークおよびライブラリーの数多くの依存関係があります。SDKをプロジェクトで適切にコンパイルするには、ビルド時にフレームワークやライブラリーにリンクするように Build Phases を再設定しなければなりません。以下は、リンクする必要のあるライブラリーです。

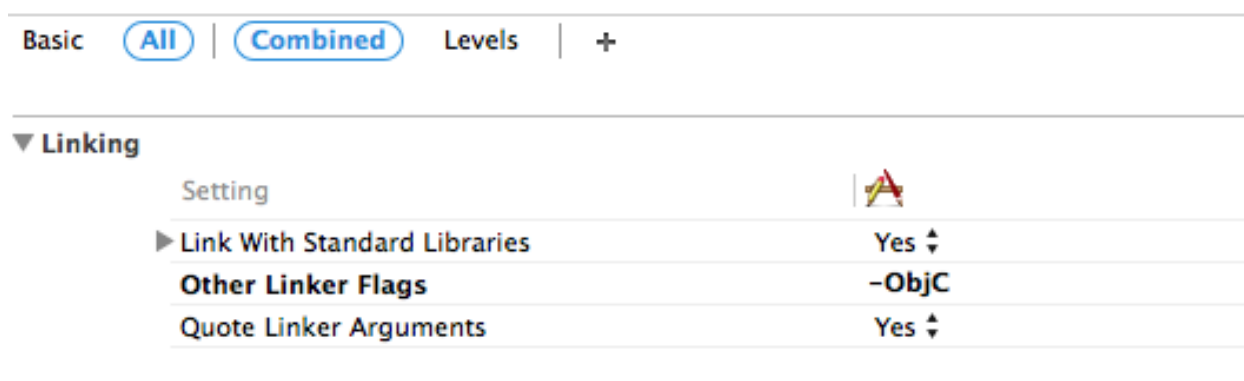
- ※ `libxml2.dylib`
- ※ `libz.dylib`
- ※ `SystemConfiguration.framework`
- ※ `CFNetwork.framework`
- ※ `MobileCoreServices.framework`

これらを追加するには、**Build Phases > Link Binary with Libraries**に移動し、上記の依存関係をリンク依存関係として追加します。

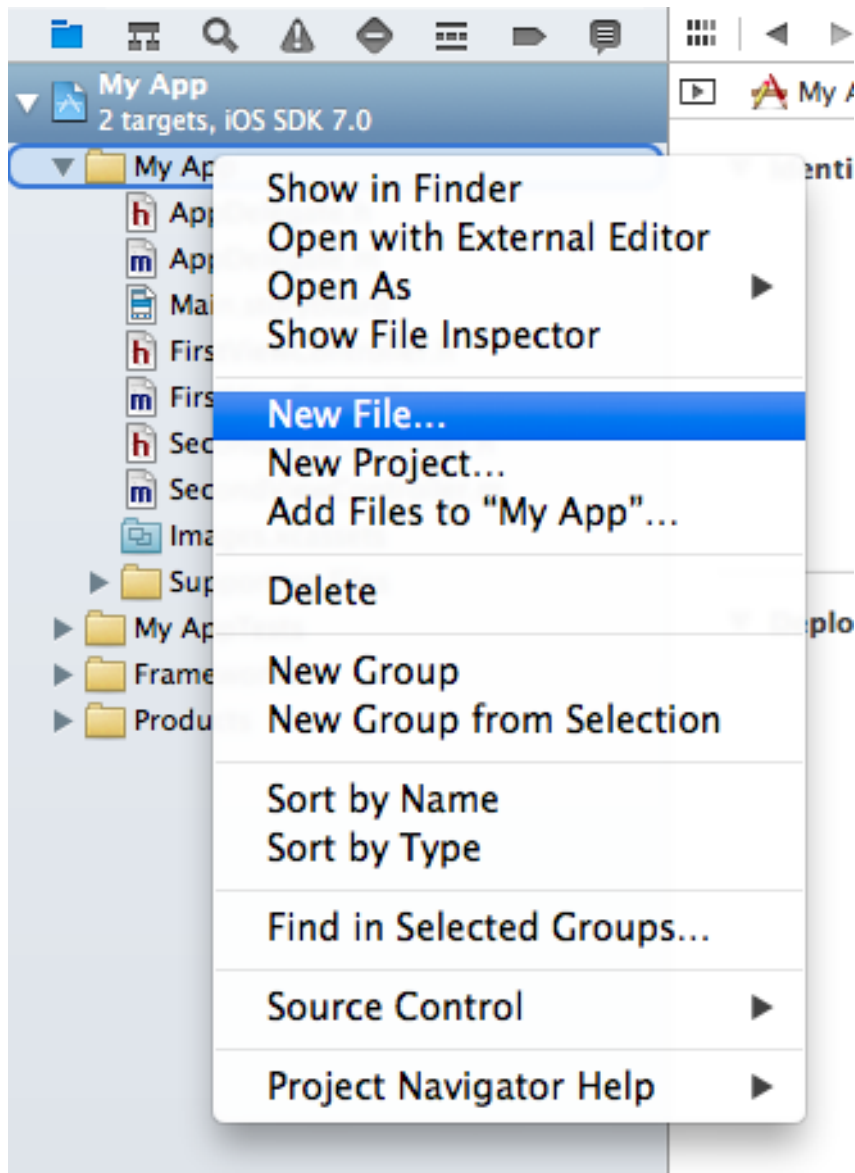


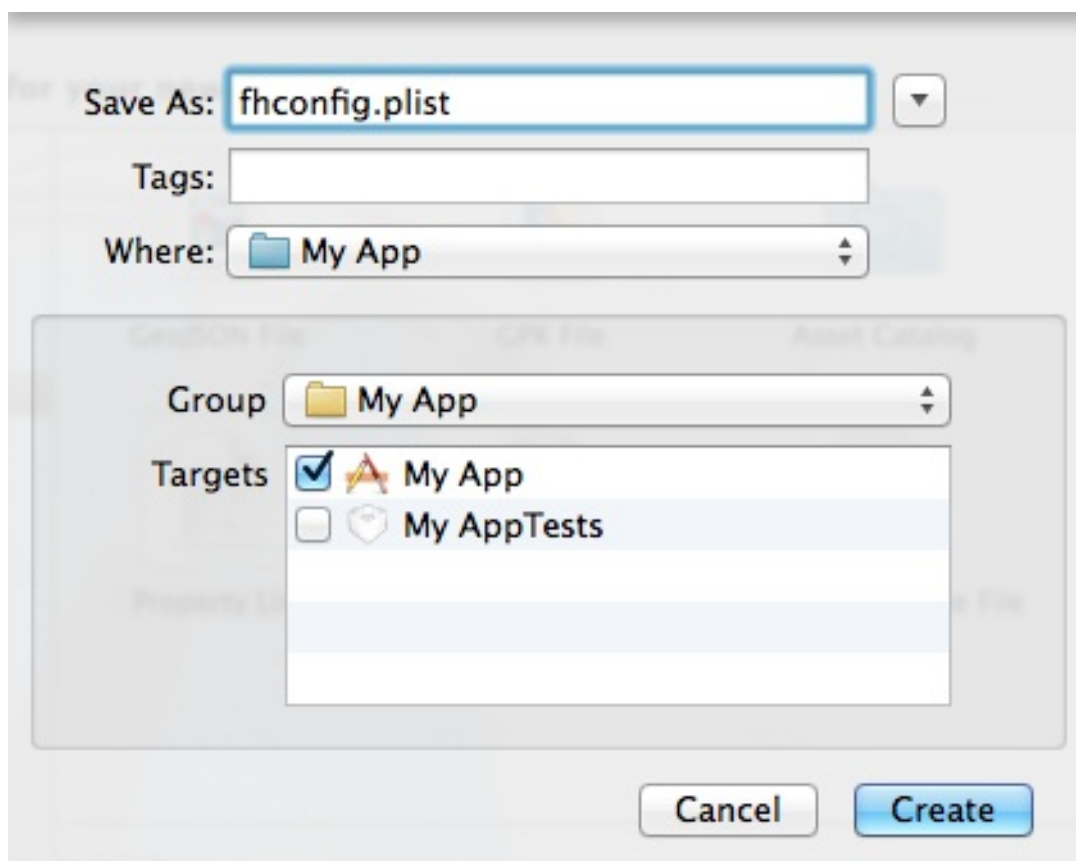
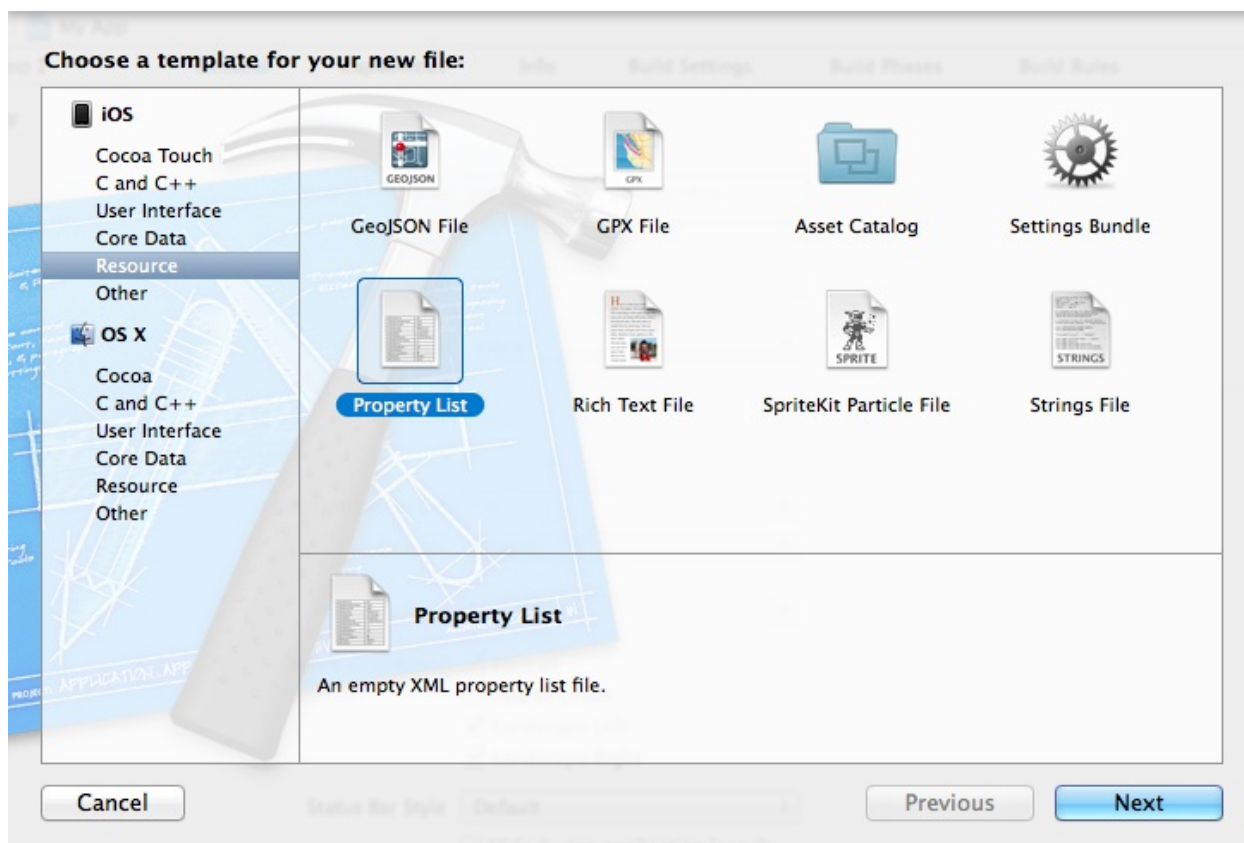
また、リンカーフラグをアプリに追加する必要があります。これを追加するには、プロジェクトのターゲットの **Build Settings** 画面に移動します。検索を実行して **Other linker flags** を見つけ、以下を追加します。

-Objective-C



最後に、**fhconfig.plist** というプロパティリストファイルをプロジェクトに追加します。





2.2.4. セットアップ

アプリが RHMAP サーバーと通信できるようにするプロパティを定義する必要があります。以下の値を **fhconfig.plist** 設定ファイルに追加します。

```
<?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>

```

クライアントアプリとクラウドアプリ間の通信についての詳細は、「[Projects - Connections](#)」を参照してください。

2.2.5. 初期化

RHMAP iOS Swift SDK からクラウド要求を呼び出す前に、初期化する必要があります。以下のコードスニペットを App Delegate の **didFinishLaunchingWithOptions:** メソッドか、またはコードがクラウド要求前に確実に呼び出されるその他の場所にコピーします。

```

#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;
}

```

2.3. API ドキュメント

[API Docs](#): すべてのクライアント API についてのドキュメント

第3章 ネイティブ iOS (SWIFT)

3.1. ダウンロード

🔗 [SDK](#)

🔗 [サンプルアプリ](#)

警告

ビルドファームを使ってビルドした Swift ベースの iOS アプリについて、enterprise distribution 証明書を使用して署名する場合、アプリが起動時にクラッシュする可能性があります。

この問題が発生した場合は、問題の解決方法の詳細について、Red Hat ナレッジベースの記事「[Swift-based iOS application crashes upon startup when signed using an enterprise distribution certificate without Organisational Unit field](#)」を参照してください。

3.2. スタート

この SDK により、iOS バージョン 8 以上の Swift アプリで RHMAP API を使用することができます。

RHMAP iOS Swift SDK は、Github の [FeedHenry iOS SDK リポジトリ](#) でホストされるオープンソースのプロジェクトです。ご自由にプロジェクトのフォークを行い、このプロジェクトに貢献してください。

この SDK を使用する前に:

- 🔗 Xcode IDE をインストールします。Xcode は [Apple 開発者ポータル](#) からダウンロードできます。
- 🔗 CocoaPods 依存関係管理システムをインストールします。CocoaPods をインストールするには以下を実行します。

```
sudo gem install cocoapods
```

詳細は、CocoaPods web サイトにある『[Getting Started guide](#)』を参照してください。

3.2.1. 新規アプリ

[サンプルアプリ](#) をクローンし、RHMAP iOS Swift SDK が CocoaPods の依存関係として組み込まれた新規 iOS アプリケーションを使って開始します。

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

[Podfile](#) で定義された依存関係を取得します。

```
pod install
```

Xcode で **blank-ios-app.xcworkspace** ワークスペースを開きます。必要な依存関係は Pods グループにあります。

3.2.2. 既存アプリ

アプリに Podfile がない場合、以下の内容の **Podfile** という新規ファイルをプロジェクトのルートに作成してください。

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

xcodproj 'ProjectName.xcodproj'
platform :ios, '8.0'
use_frameworks!

pod 'FeedHenry', '4.0.0'
```

4.0.0 をターゲットにしている RHMAP iOS Swift SDK のバージョンに、**ProjectName.xcodproj** をプロジェクトの名前に置き換えます。バージョン番号を指定しない場合、CocoaPods 中央リポジトリの最新バージョンが使用されます。

Podfile で定義された依存関係を取得します。

```
pod install
```

Xcode で **ProjectName.xcworkspace** を開けるようになります。

3.2.3. セットアップ

アプリが RHMAP サーバーと通信できるようにするプロパティを定義する必要があります。以下の値を **fhconfig.plist** 設定ファイルに追加します。

```
<?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>
```

クライアントアプリとクラウドアプリ間の通信についての詳細は、「[Projects - Connections](#)」を参照してください。

3.2.4. 初期化

RHMAP iOS Swift SDK からクラウド要求を呼び出す前に、初期化する必要があります。以下のコードスニペットを App Delegate の **application(_ , didFinishLaunchingWithOptions:)** メソッドか、またはコードがクラウド要求前に確実に呼び出されるその他の場所にコピーします。

```
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 ドキュメント

[API Docs](#): すべてのクライアント API についてのドキュメント

第4章 ネイティブ WINDOWS

4.1. ダウンロード

📄 [SDK](#)

📄 [サンプルアプリ](#)

4.2. はじめに

これは、標準的な Windows Phone のネイティブアプリです。

この SDK 自体は、[こちら](#)でホストされるオープンソースプロジェクトです。ご自由にプロジェクトのフォークを行い、このプロジェクトに貢献してください。

この SDK を使用する前に、Windows Phone Developer Tools がインストールされていることを確認してください。これらは、[こちらから](#)ダウンロードできます。

4.3. 新規アプリ

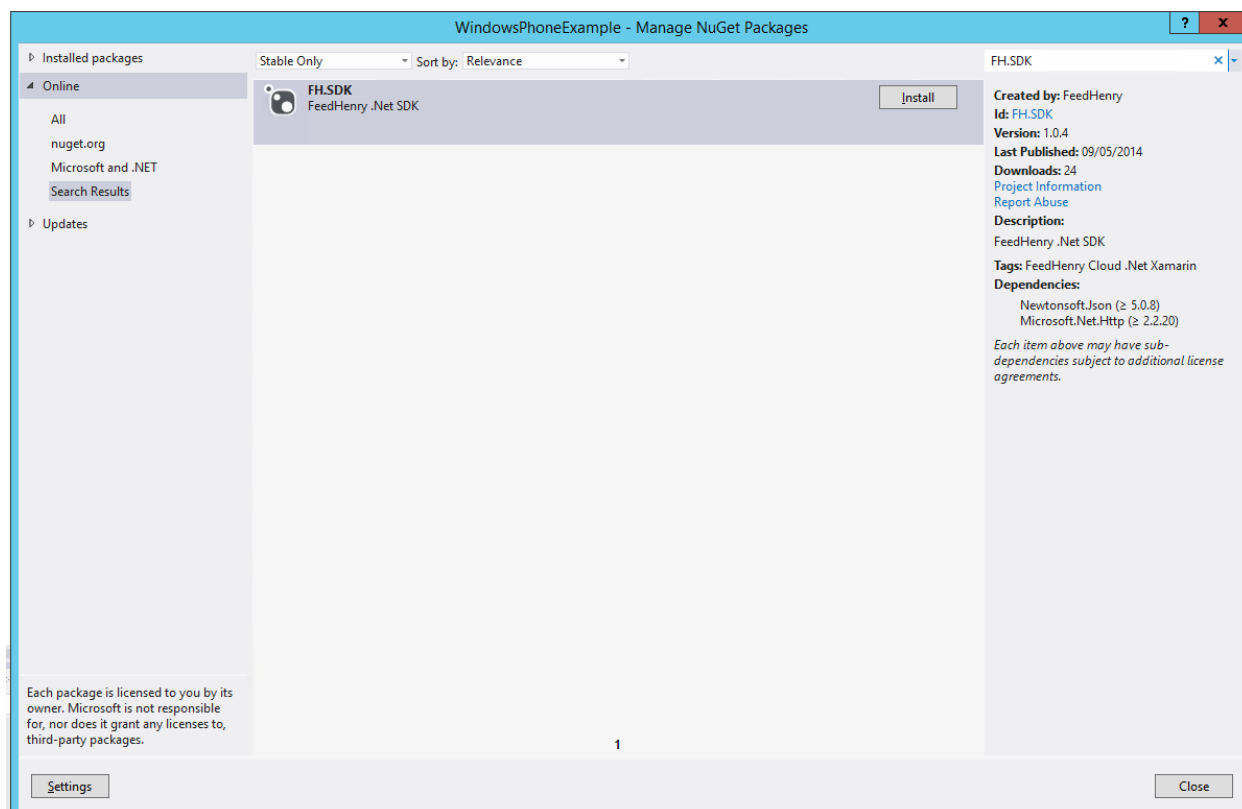
[サンプルアプリ](#) をダウンロードし、すでに RHMAP SDK が組み込まれている新規の Windows Phone アプリを使って開始します。

4.4. 既存アプリ

SDK のプロジェクトへのインストールは、(NuGet を使用して) 自動で行うことも、手動で行うこともできます。

4.4.1. NuGet (推奨)

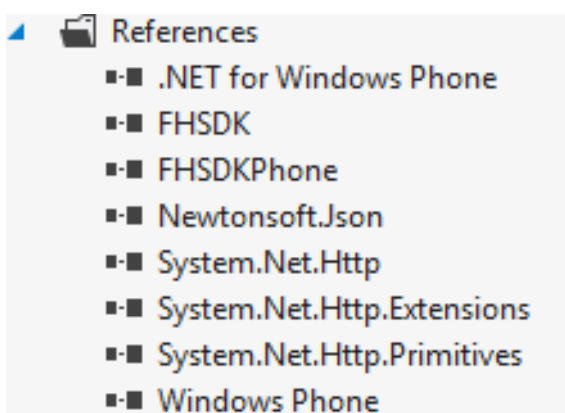
FH SDK は NuGet (<https://www.nuget.org/packages/FH.SDK/>) から利用できます。Visual Studio 内で NuGet プラグインを使用している場合は、FH.SDK を検索してください。NuGet は依存関係のライブラリーを自動的にインストールします。



4.4.2. 手動

SDK をダウンロードし、これを展開します。**.dll** アセンブリファイルを、プロジェクトの参照として **wp80** フォルダー内に追加します。

Name	Date modified	Type	Size
FHSDK.dll	12/05/2014 18:42	Application extens...	32 KB
FHSDK.xml	12/05/2014 18:42	XML File	36 KB
FHSDKPhone.dll	12/05/2014 18:42	Application extens...	12 KB
FHSDKPhone.xml	12/05/2014 18:42	XML File	3 KB



Portable Class Library プロジェクトを開発している場合は、**FHSDK.dll** ファイルのみを参照します。

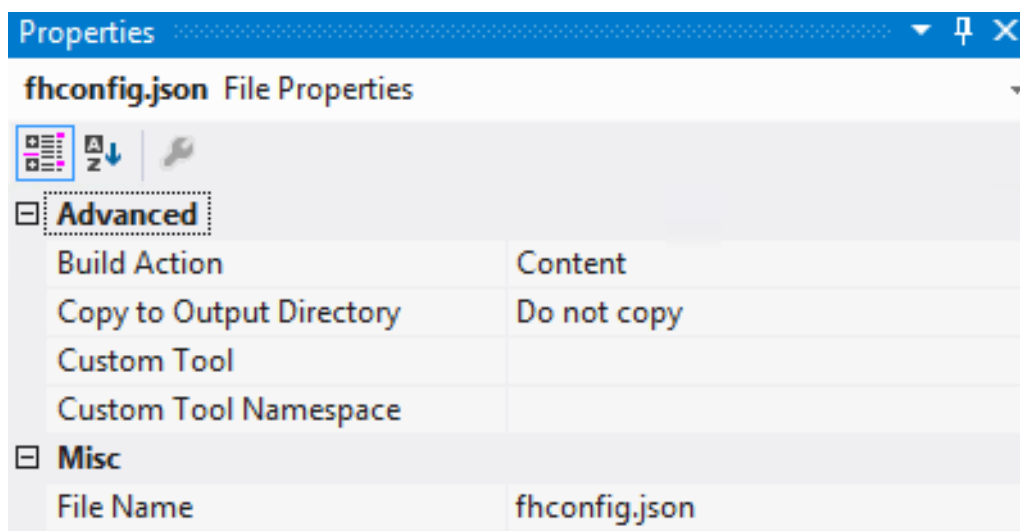
この SDK は、[Json.Net](#) および [Microsoft HTTP クライアントライブラリー](#) に依存しています。これらのライブラリーのアセンブリーも、プロジェクトで利用可能でない場合はインストールする必要があります。

4.4.3. 設定のセットアップ

プロジェクトに **fhconfig.json** という新規ファイルを作成する必要があります。ファイルの内容は以下のようになります。

```
{
  "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__"
}
```

ファイルの Build Action が **Content** であることを確認します。



4.4.4. 初期化

RHMAP .NET SDK を使用するには、(通常はアプリの起動時に) 以下のように SDK を初期化する必要があります。

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

接続についての詳細は、[こちら](#)を参照してください。

注記

Init メソッドのみが **FHClient** クラスを使用して呼び出され、かつプラットフォーム固有のプロジェクトから呼び出す必要のあるメソッドです (例: [PCL プロジェクト](#)からは呼び出すことができません)。

これ以外のすべての SDK メソッドは、**FHSDK.dll** アセンブリーで定義される **FH** クラスを使用して呼び出されます。このアセンブリーは他の PCL プロジェクトから参照することができます。このようにクロスプラットフォームソリューションに PCL プロジェクトが含まれる場合には、このアセンブリーファイルを参照し、そこから SDK 機能呼び出すことができます。

4.5. HTTPCLIENT の独自オプションの使用

デフォルトで、.NET SDK は [Microsoft HTTP クライアントライブラリ](#) を使用してすべての HTTP 要求を実行します。ただし、Xamarin を使用して iOS と Android アプリを開発している場合には、[ModernHttpClient](#) を選択する方が良いでしょう。これを使用するには、アプリに **ModernHttpClient** コンポーネントをインストールすることのみが必要であり、以下のようにこれを使用します。

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

上記のいずれも使用することを望まない場合は、いずれかの HTTP (または REST) を使用できます。この場合、必要になるのはアプリのクラウドホストのみで、以下のメソッドを使って取得できます。

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

ただし、この方法には、一部のデータが要求内に欠落するため、プラットフォームによって提供される分析サービスをアプリで使用できなくなるという不利な点があります。分析サービスを再度有効にするには、以下のメソッドで返されるメタデータを、ヘッダーのセットとして各 HTTP 要求に追加することのみが必要になります。

```
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. SDK の使用

SDK 内で利用可能な API についての詳細は [API Docs](#) を参照してください。

第5章 XAMARIN

5.1. ダウンロード

※ [SDK](#)

※ [サンプルアプリ](#)

5.2. はじめに

これは、ネイティブ iOS、Android および Windows Phone アプリを C# で作成することを可能にする標準的な Xamarin アプリです。

この SDK 自体は、[こちら](#)でホストされるオープンソースプロジェクトです。ご自由にプロジェクトのフォークを行い、このプロジェクトに貢献してください。

この SDK を使用する前に、Xamarin 開発者ツールがインストールされていることを確認してください。これらは、[こちらから](#)ダウンロードできます。

Windows Phone アプリを開発するには、Windows Phone Developer Tools をインストールしておく必要があります。これは、[こちらから](#)ダウンロードできます。

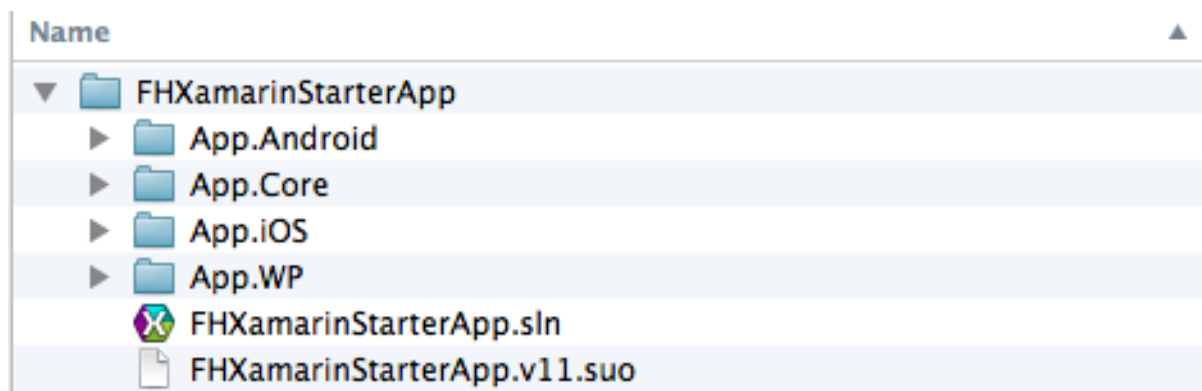
[Xamarin For Visual Studio](#) をインストールすることもお勧めします。

5.3. 新規アプリ

[サンプルアプリ](#)をダウンロードし、すでに RHMAR SDK が組み込まれている新規の Xamarin アプリを使って開始します。

このアプリには 4 つのサブプロジェクトが含まれます。このアプリは、すべてのアプリでコードを共有するために Portable Class Library を使用できるようにセットアップされています。この方法についての詳細は、[こちら](#)を参照してください。

- ※ **App.Core**: PCL プロジェクトです。このプロジェクトのコードは他のアプリによって共有されます。アプリのビジネスロジックのほとんどをここで定義する必要があります。
- ※ **App.WP**: Windows Phone アプリプロジェクトは **App.Core** プロジェクトに依存します。通常、これには UI コードと WP 固有のコードが含まれます。
- ※ **App.Android**: Android アプリプロジェクトは **App.Core** プロジェクトに依存します。通常、これには UI コードと Android 固有のコードが含まれます。
- ※ **App.iOS**: iOS アプリプロジェクトは **App.Core** プロジェクトに依存します。通常、これには UI コードと iOS 固有のコードが含まれます。

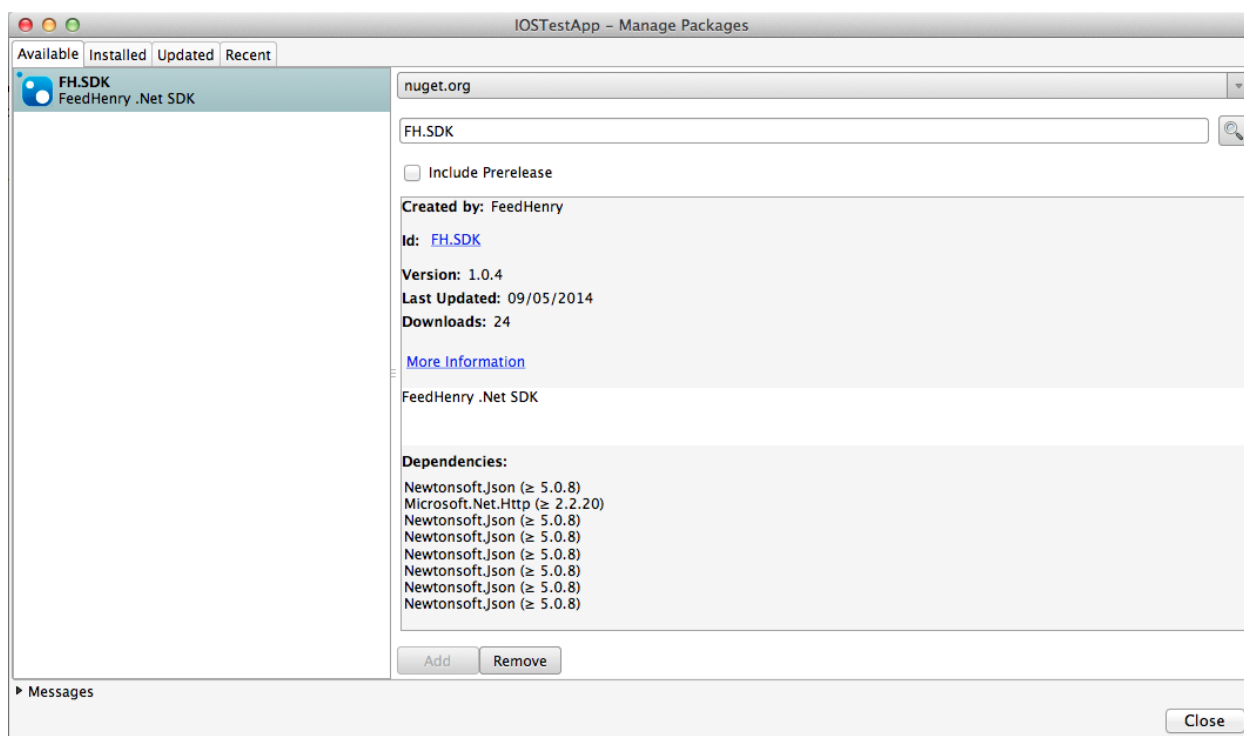


5.4. 既存アプリ

SDK のプロジェクトへのインストールは、(NuGet を使用して) 自動で行うことも、手動で行うこともできます。

5.4.1. NuGet (推奨)

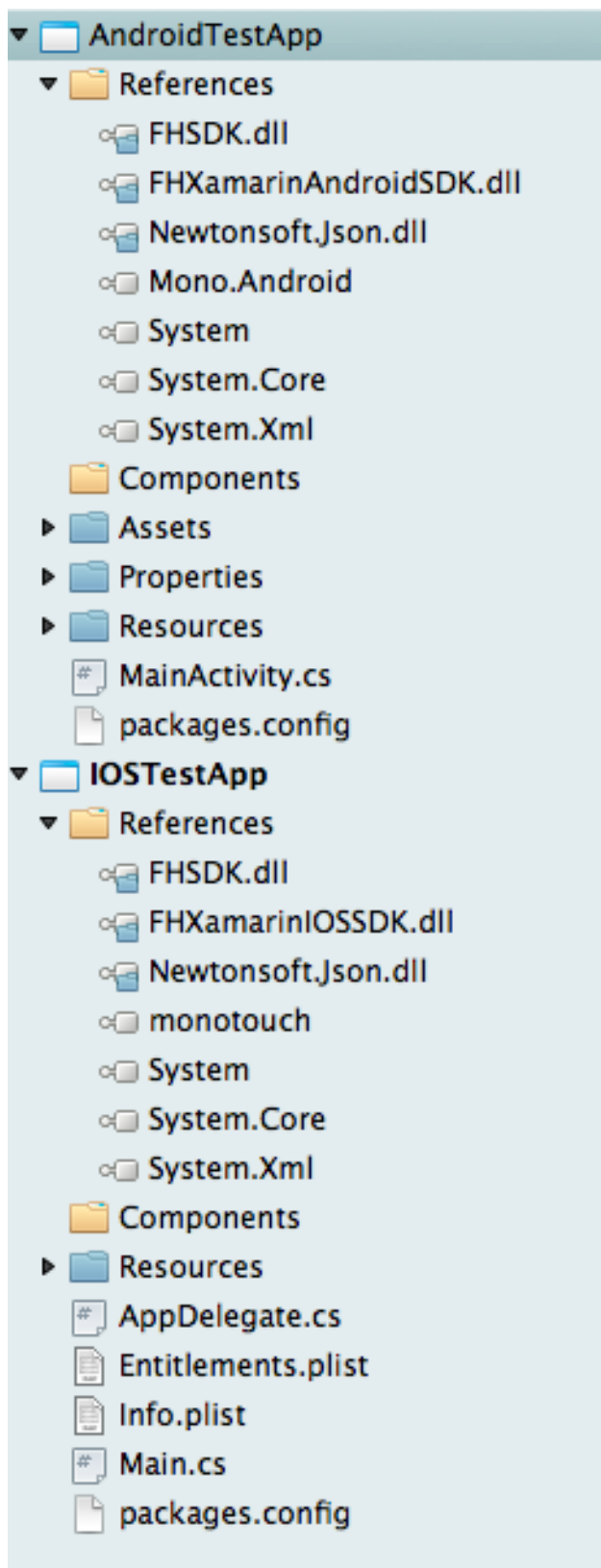
FH SDK は NuGet (<https://www.nuget.org/packages/FH.SDK/>) から利用できます。Xamarin Studio 内で NuGet プラグインを使用している場合は、FH.SDK を検索してください。NuGet は依存関係のライブラリーを自動的にインストールします。



5.4.2. 手動

SDK をダウンロードし、これを展開します。プロジェクトのビルドターゲットに対応するフォルダーから **.dll** アセンブリファイルを参照として追加します。

Name		
▼	monoandroid	
	FHSDK.dll	
	FHSDK.xml	
	FHXamarinAndroidSDK.dll	
	FHXamarinAndroidSDK.XML	
▼	monotouch	
	FHSDK.dll	
	FHSDK.xml	
	FHXamarinIOSSDK.dll	
	FHXamarinIOSSDK.XML	
▼	wp80	
	FHSDK.dll	
	FHSDK.xml	
	FHSDKPhone.dll	
	FHSDKPhone.xml	



Portable Class Library プロジェクトを開発している場合は、**FHSDK.dll** ファイルのみを参照します。

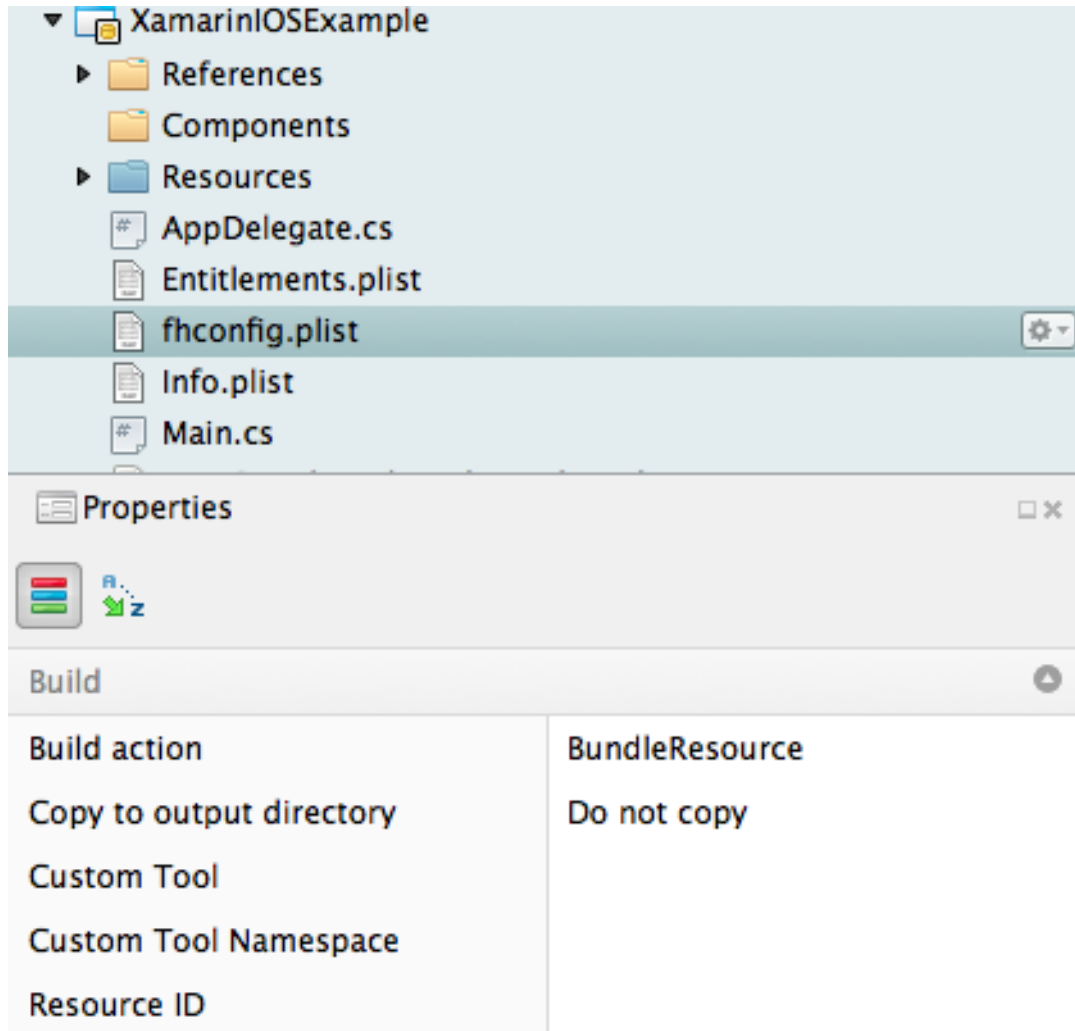
この SDK は、[Json.Net](#) および [Microsoft HTTP クライアントライブラリー](#) に依存しています。これらのライブラリーのアセンブリーも、プロジェクトで利用可能でない場合はインストールする必要があります。

5.4.3. 設定のセットアップ

プラットフォーム固有のそれぞれのアプリケーションについて、対応する設定ファイルを作成する必要があります。それぞれのファイルの内容は、各プラットフォームのネイティブ SDK doc に記載される内容と同じである必要があります。

5.4.4. iOS

fhconfig.plist がアプリケーションのルートにあります。Build Action を **BundleResource** に設定します。



fhconfig.plist ファイルの内容:

```
<?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>
```

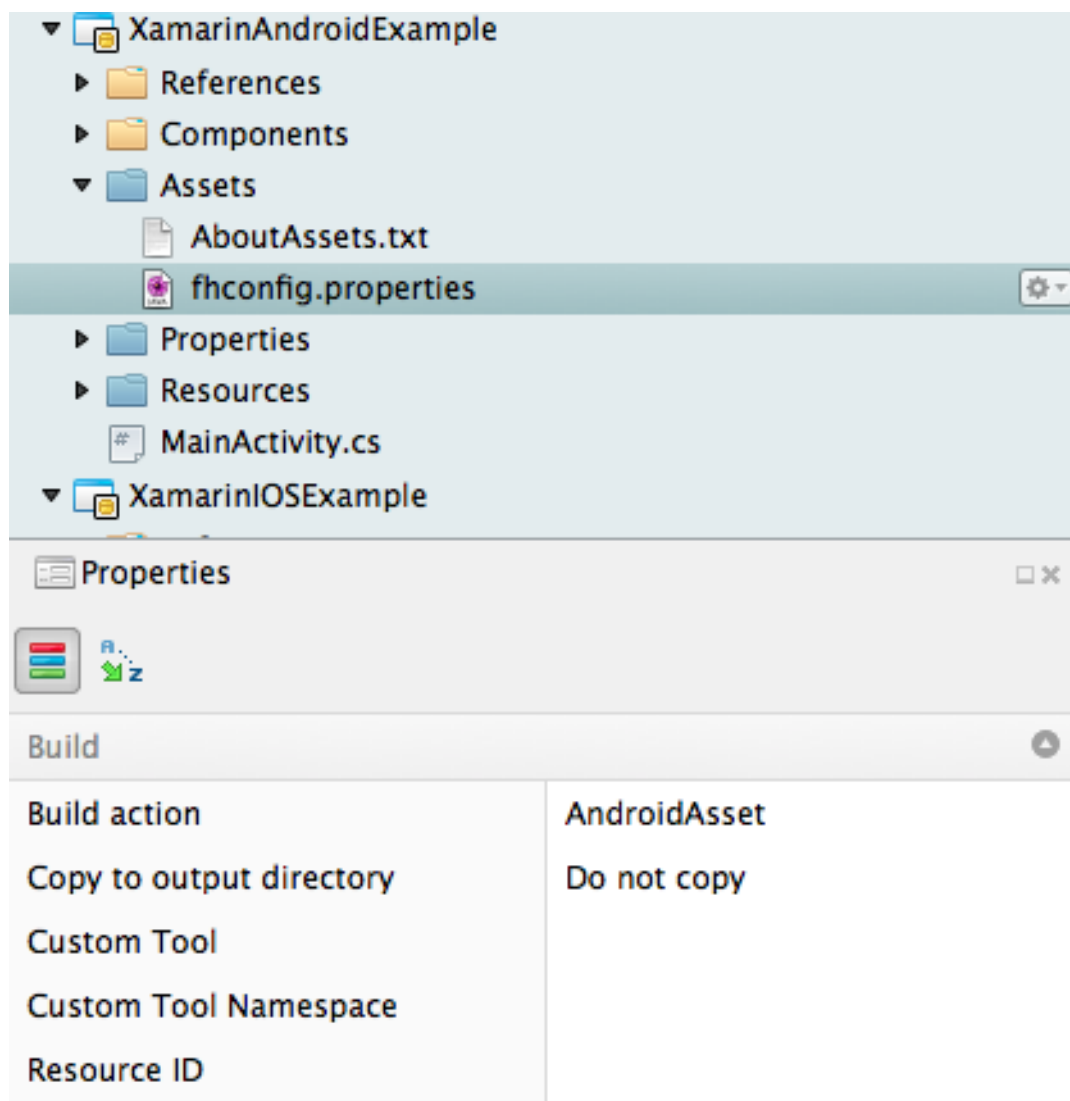
```

    <key>connectiontag</key>
    <string>__CONNECTION_TAG_TO_USE_FOR_CLOUD__</string>
  </dict>
</plist>

```

5.4.5. Android

fhconfig.properties がアプリケーションの **Assets** ディレクトリーにあります。Build Action を **AndroidAsset** に設定します。



fhconfig.properties ファイルの内容:

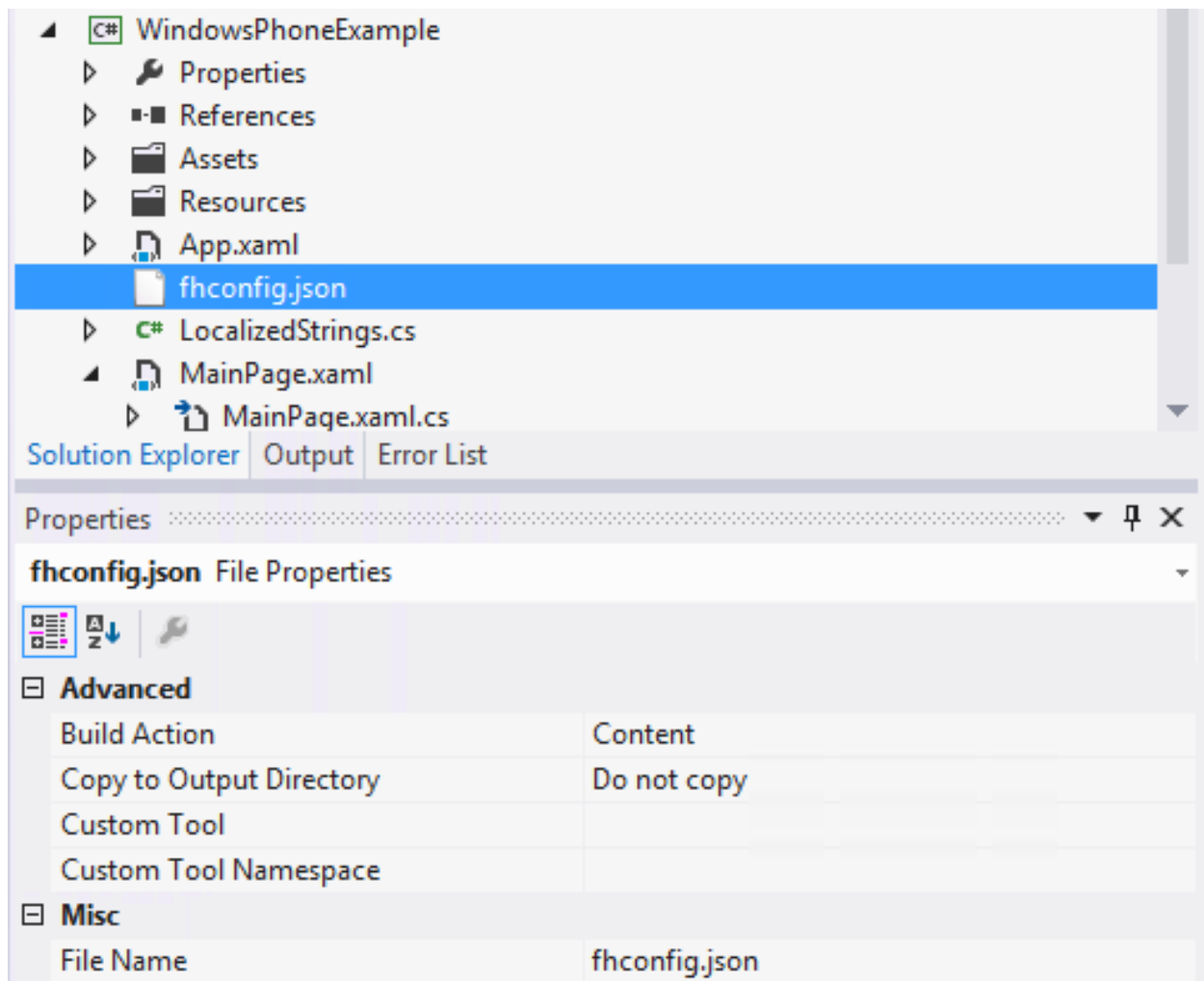
```

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__

```

5.4.6. Windows Phone 8

fhconfig.json がアプリケーションのルートにあります。Build Action を **Content** に設定します。



fhconfig.json ファイルの内容:

```
{
  "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__"
}
```

接続についての詳細は、[こちら](#)を参照してください。

5.4.7. 初期化

RHMAP .NET SDK を使用するには、アプリの起動が終了する際に、プラットフォーム固有のプロジェクト (PCL プロジェクトではない) で以下のように SDK を初期化する必要があります。

```
try
{
    bool initd = await FHClient.Init();
    if(initd) {
        //Initialisation is successful
    }
}
```

```
catch(FHException e)
{
    //Initialisation failed, handle exception
}
```

FHClient は以下のネームスペースで利用できます。

✧ **FHSDK.Phone**: WP8

✧ **FHSDK.Droid**: Android

✧ **FHSDK.Touch**: iOS

アプリのビルドターゲットに応じ、上記のネームスペースの 1 つのみをアプリで利用可能にする必要があります。

複数の異なるネームスペースで同じ **FHClient** クラスを定義するのは、主にプラットフォーム固有のアセンブリーファイルを適切にロードする必要があるためです。

注記

Init メソッドのみが **FHClient** クラスを使用して呼び出され、かつプラットフォーム固有のプロジェクトから呼び出す必要のあるメソッドです (例: PCL プロジェクトからは呼び出すことができません)。

これ以外のすべての SDK メソッドは、**FHSDK.dll** アセンブリーで定義される **FH** クラスを使用して呼び出されます。このアセンブリーは他の PCL プロジェクトから参照することができます。このようにクロスプラットフォームソリューションに PCL プロジェクトが含まれる場合、このアセンブリーファイルを参照し、そこから SDK 機能呼び出すことができます。

5.5. HTTPCLIENT の独自オプションの使用

デフォルトで、.NET SDK は [Microsoft HTTP クライアントライブラリ](#) を使用してすべての HTTP 要求を実行します。ただし、Xamarin を使用して iOS と Android アプリを開発している場合には、[ModernHttpClient](#) を選択する方が良いでしょう。これを使用するには、アプリに ModernHttpClient コンポーネントをインストールすることのみが必要であり、以下のようにこれを使用します。

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

上記のいずれも使用することを望まない場合は、いずれかの HTTP (または REST) を使用できます。この場合、必要になるのはアプリのクラウドホストのみで、以下のメソッドを使って取得できます。

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

ただし、この方法には、一部のデータが要求内に欠落するため、プラットフォームによって提供される分析サービスをアプリで使用できなくなるという不利な点があります。分析サービスを再度有効にするには、以下のメソッドで返されるメタデータを、ヘッダーのセットとして各 HTTP 要求に追加することのみが必要になります。


```
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);
}
...
```

5.6. SDK の使用

SDK 内で利用可能な API についての詳細は [API Docs](#) を参照してください。

第6章 CORDOVA

6.1. ダウンロード

🔗 [SDK](#)

🔗 [サンプルアプリ](#)

6.2. スタート

RHMAP JavaScript SDK により、Cordova アプリで RHMAP API を使用することができます。このアプリは主に HTML、JS、CSS などの Web テクノロジーを使用して開発され、また数多くのデバイス機能へのアクセスを許可します。

基礎となるネイティブプロジェクトおよび Cordova ライブラリーは開発者に公開されます。これにより、Cordova プラグインやサードパーティーライブラリーを含む、アプリケーションの完全なカスタマイズが可能になります。通常、Cordova アプリのネイティブコードの作成または修正に必要な時間は比較的短く、開発チーム内の小規模なグループのみがネイティブコードを扱えるようにするだけで十分です。この小規模なグループは、ネイティブコードを処理したり、Cordova プラグインアーキテクチャーを使用して追加のプラグインまたは SDK を JavaScript API で公開したりできます。

Cordova についての詳細は、[公式 Cordova web サイト](#) を参照してください。

6.2.1. 新規アプリ

[サンプルアプリ](#) をダウンロードし、すでに RHMAP JavaScript SDK が組み込まれている新規の Cordova アプリを使って開始します。

6.2.2. 既存アプリ

SDK をダウンロードし、**feedhenry.js** として保存します。

feedhenry.js を Cordova アプリの **index.html** ファイルと同じ場所 (通常は **www** ディレクトリー内) にコピーします。

以下のコードを **index.html** ファイルに追加します。

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

6.2.3. セットアップ

最後に、アプリが RHMAP サーバーと通信できるようにするプロパティーを定義する必要があります。SDK ファイルと同じディレクトリーに、以下の内容の **fhconfig.json** という新規ファイルを作成します。括弧内の参照部分はプロジェクトの値に置き換えます。

```
{
  "host": "<RHMAP Core host>",
  "projectid": "<Project ID>",
  "connectiontag": "<Connection tag>",
```

```

    "appid": "<Client App ID>",
    "appkey": "<Client App API key>"
  }

```

クライアントアプリとクラウドアプリ間の通信についての詳細は、「[Projects - Connections](#)」を参照してください。

6.3. 互換性

Cordova アプリ開発用のセットアップを機能させるには、ローカルの Cordova CLI バージョンが RHMAP ビルドファームにインストールされた Cordova のバージョンと一致しており、Cordova プラットフォームのバージョンがサポートされている範囲内であることを確認するようお勧めします。このセクションでは、ビルドファームでサポートされている Cordova CLI およびプラットフォームのバージョンを一覧表示します。

Cordova プラットフォームのバージョンをアプリに指定しない場合は、ビルドファームで使用する Cordova CLI のデフォルトの固定バージョンが使用されます。ビルドの再現性および安定性を確保するには、常にプラットフォームのバージョンを Cordova アプリに指定することをお勧めします。

6.3.1. 現行バージョン

Cordova CLI: **5.2.0**

プラットフォーム	最小のバージョン	最大のバージョン
Android	3.1.0	5.0.0
iOS	3.1.0	4.1.0
Windows	3.1.0	3.8.2

6.4. API ドキュメント

✳ [API Docs](#): すべてのクライアント API についてのドキュメント

第7章 CORDOVA LIGHT

7.1. ダウンロード

✎ [SDK](#)

✎ [サンプルアプリ](#)

7.2. スタート

RHMAP JavaScript SDK により、Cordova Light アプリで RHMAP API を使用することができます。Cordova Light は、**www** ディレクトリーのみを公開する Cordova アプリです。これらのアプリは、HTML、CSS、JavaScript などの標準 web テクノロジーのみを使用します。ネイティブコードは使用されません。

Cordova Light アプリの使用は、Red Hat Mobile Application Platform ホスト型 (RHMAP) の使い方を習得する上で簡単かつ分かりやすい方法となりますが、開発者は標準的な [Cordova プラグイン](#) の使用に制限されます。

7.2.1. 新規アプリ

[サンプルアプリ](#) をダウンロードし、すでに RHMAP SDK が組み込まれている新規の Cordova Light アプリを使って開始します。

7.2.2. 既存アプリ

SDK をダウンロードし、**feedhenry.js** として保存します。

feedhenry.js を Cordova アプリの index.html ファイルと同じ場所 (通常は /**www** ディレクトリー内) にコピーします。

以下のコードを **index.html** ファイルに追加します。

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

7.2.3. セットアップ



注記

fhconfig.json ファイルは /**www** ディレクトリーにすでに存在している可能性があります。この場合は、このファイルの内容の確認のみが必要になります。

最後に、アプリが RHMAP サーバーと通信できるようにするプロパティーを定義する必要があります。SDK ファイルと同じディレクトリーに、以下の内容の **fhconfig.json** という新規ファイルを作成します。括弧内の参照部分はプロジェクトの値に置き換えます。

```
{
  "host": "<RHMAP Core host>",
```

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

クライアントアプリとクラウドアプリ間の通信についての詳細は、「[Projects - Connections](#)」を参照してください。

7.3. API ドキュメント

✳ [API Docs](#): すべてのクライアント API についてのドキュメント

第8章 WEB

8.1. ダウンロード

🔗 [SDK](#)

🔗 [サンプルアプリ](#)

8.2. はじめに

これは、Node.js + Express web アプリケーションです。これらのアプリはより高度なデスクトップ/タブレットの web ポータルおよびモバイル web サイトを提供します。

また、[Express 4](#) や [ejs](#) などのテンプレートエンジンを使用したサーバーサイドのテンプレートなどの機能を含む、web アプリ開発のための Node.js の完全な機能を公開します。

標準的な HTML5、CSS および JavaScript の静的ファイルサービスもサポートします。

8.3. 新規アプリ

[サンプルアプリ](#) をダウンロードし、すでに RHMAP SDK が組み込まれている新規の Web アプリを使って開始します。

8.4. 既存アプリ

Web Apps アプリは、**index.html** ファイルに組み込まれている標準的な JavaScript SDK を使用します。

8.4.1. SDK のダウンロードおよびコピー

SDK をダウンロードし、**feedhenry.js** として保存します。

上記のダウンロードされた JavaScript ファイルを Web App にコピーし、参照できるようにします。このファイルは、**index.html** ファイルと同じ場所 (通常は **/public** ディレクトリー内) にコピーする必要があります。

8.4.2. SDK の統合

以下のコードを **index.html** ファイルに追加します。

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

8.4.3. 設定のセットアップ

SDK ファイル (**feedhenry.js**) と同じディレクトリーに、以下の内容の **fhconfig.json** という新規ファイルを作成します。

```
{
```

```
"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__"  
}
```

接続についての詳細は、[こちら](#)を参照してください。

8.5. SDK の使用

SDK 内で利用可能な API についての詳細は [API Docs](#) を参照してください。

第9章 APPCELERATOR

9.1. ダウンロード

🔗 [SDK](#)

🔗 [サンプルアプリ](#)

9.2. はじめに

これは、標準的な Appcelerator Titanium アプリです。

この SDK を使用する前に、Titanium 開発者ツールがインストールされていることを確認してください。これらは、[こちらから](#)ダウンロードできます。

9.3. 新規アプリ

[サンプルアプリ](#) をダウンロードし、すでに RHMAP SDK が組み込まれている新規の Appcelerator Titanium アプリを使って開始します。

9.4. 既存アプリ

Appcelerator Titanium アプリは標準的な JavaScript SDK の 1 つの種類を使用します。

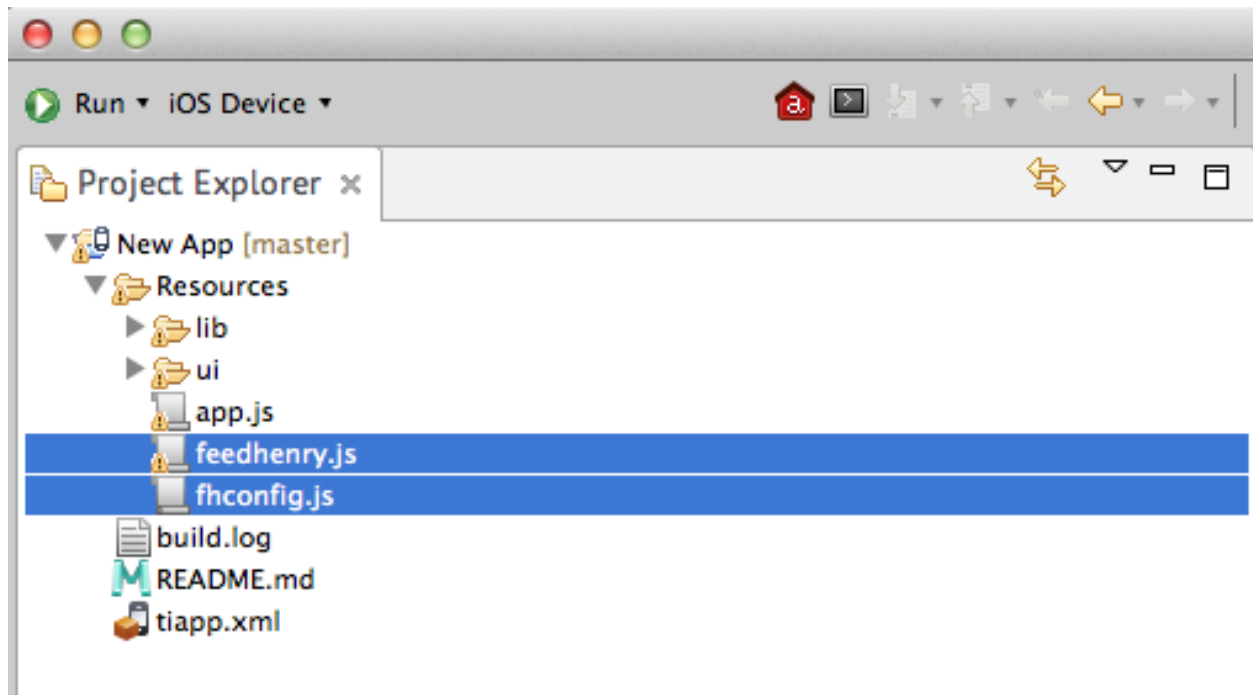
9.4.1. SDK のダウンロードおよびコピー

SDK をダウンロードし、feedhenry.js として保存します。

上記のダウンロードされた JavaScript ファイルを Basic Web アプリにコピーし、参照できるようにします。このファイルは、**index.html** ファイルと同じ場所 (通常は /**www** ディレクトリー内) にコピーする必要があります。

9.4.2. SDK の統合

SDK を統合するには、上記のダウンロードされた JavaScript ファイルを参照できるように /**Resources** ディレクトリーの Appcelerator プロジェクトにコピーする必要があります。



9.4.3. 設定のセットアップ

SDK ファイル (**feedhenry.js**) と同じディレクトリーに、以下の内容の **fhconfig.js** という新規ファイルを作成します。

```
exports = {  
  "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__"  
};
```

接続についての詳細は、[こちら](#)を参照してください。

9.5. SDK の使用

SDK 内で利用可能な API についての詳細は [API Docs](#) を参照してください。