Migrating existing iOS applications

Overview

To migrate an existing native iOS project that was created with IBM MobileFirst™ Platform Foundation version 6.2.0 or later, you must modify the project to use the SDK from the current version. Then you replace the client-side APIs that are discontinued or not in v8.0. The migration assistance tool can scan your code and generate reports of the APIs to replace.

Jump to

Scanning existing MobileFirst native iOS apps to prepare for MobileFirst version 8.0

The migration assistance tool helps you prepare your apps that were created with previous versions of IBM MobileFirst™ Platform Foundation for migration by scanning the sources of the native iOS apps that were developed by using Swift or Objective-C and generating a report of APIs that are deprecated or discontinued in version 8.0.

The following information is important to know before you use the migration assistance tool:

- You must have an existing IBM MobileFirst Platform Foundation native iOS application.
- You must have internet access.
- You must have node.js version 4.0.0 or later installed.
- Review and understand the limitations of the migration process. For more information, see Migrating apps from earlier releases (../).

Apps that were created with earlier versions of IBM MobileFirst Platform Foundation are not supported in IBM MobileFirst Platform Foundation version 8.0 without some changes. The migration assistance tool simplifies the process by scanning the source files in the existing version app and identifies APIs that are deprecated, no longer supported, or modified in version 8.0.

The migration assistance tool does not modify or move any developer code or comments of your app.

1. Download the migration assistance tool by using one of the following methods:
   - Download the .tgz file from the Jazzhub repository (https://hub.jazz.net/project/ibmmfpf/mfp-migrator-tool).
   - Download the Developer Kit, which contains the migration assistance tool as a file named **mfpmigrate-cli.tgz**, from the MobileFirst Operations Console.
2. Install the migration assistance tool.
   - Change to the directory where you downloaded the tool.
   - Use NPM to install the tool by entering the following command:

   ```
   npm install -g
   ```

3. Scan the IBM MobileFirst Platform Foundation app by entering the following command:

   ```
   mfpmigrate scan --in source_directory --out destination_directory --type ios
   ```

   source_directory
   The current location of the version project.
   destination_directory
   The directory where the report is created.

   When it is used with the scan command, the migration assistance tool identifies APIs in the existing IBM MobileFirst Platform Foundation app that are removed, deprecated, or changed in version 8.0 and saves them in the identified destination directory.

Migrating an existing iOS project to version 8.0 manually

Migrate your existing native iOS project manually within your Xcode project and continue developing with IBM MobileFirst Foundation V8.0.

Before you begin you must:

- be working in Xcode 7.0 (iOS 9) or later.
- have an existing native iOS project that was created with IBM MobileFirst Platform Foundation 6.2.0 or later.
- have access to a copy of the V8.0.0 MobileFirst iOS SDK files.

1. Delete all of the existing references to the static library **libWorklightStaticLibProjectNative.a** in the **Link Binary With Libraries** tab of **Build Phases** section.
2. Delete the Headers folder from the **WorklightAPI** folder.
3. In the **Build Phases** section, link the main required framework **IBMMobileFirstPlatformFoundation.framework** file in the **Link Binary With Libraries** tab.
   This framework provides core MobileFirst functionality. Similarly, you can add other frameworks for optional functionality (../../../application-development/sdk/ios/#manually-adding-the-mobilefirst-native-sdk).
4. Similar to the preceding step, link the following resources to your project in the **Link Binary With Libraries** section of the **Build Phases** tab.
   - SystemConfiguration.framework
   - MobileCoreServices.framework
   - Security.framework
   - Note: Some frameworks might already be linked.
     - libstdc++.6.tbd
     - libz.tbd
     - libc++.tbd
5. Remove **$(SRCROOT)/WorklightAPI/include** from the header search path.

6. Replace all of the existing MobileFirst imports of headers with a single entry of the following new umbrella header:
    - Objective-C:

      ```
      #import <IBMMobileFirstPlatformFoundation/IBMMobileFirstPlatformFoundation.h>
      ```

    - Swift:

      ```
      import IBMMobileFirstPlatformFoundation
      ```

Your application is now upgraded to work with the IBM MobileFirst Foundation, v8.0 iOS SDK.

What do to next

Replace the client-side APIs that are discontinued or not in v8.0.

Migrating an existing native iOS projects to version 8.0 with CocoaPods

Migrate your existing native iOS project to work with v8.0 by getting the IBM MobileFirst Foundation iOS SDK using CocoaPods and making changes in the project configuration.

> **Note:** MobileFirst development is supported in Xcode from version 7.1 by using iOS 8.0 and later.

You must have:

- CocoaPods installed in your development environment.
- Xcode 7.1 with iOS 8.0 or higher for your development environment.
- An app integrated with MobileFirst 6.2 or later.

The SDK contains required and optional SDKs. Each required or optional SDK has its own pod.
The required IBMMobileFirstPlatformFoundation pod is the core of the system. It implements client-to-server connections, handles security, analytics, and application management.

The following optional pods provide additional features.

Pod	Feature
IBMMobileFirstPlatformFoundationPush	Adds the IBMMobileFirstPlatformFoundationPush framework for enabling Push.
IBMMobileFirstPlatformFoundationJSONStore	Implements the JSONStore feature. Include this pod in your Podfile if you intend to use the JSONStore feature in your app.
IBMMobileFirstPlatformFoundationOpenSSLUtils	Contains the MobileFirst embedded OpenSSL feature and loads automatically the openssl framework. Include this pod in your Podfile if you intend to use the OpenSSL provided by MobileFirst.

1. Open your project in Xcode.
2. Delete the **WorklightAPI** folder from your Xcode project (move it to trash).
3. Modify your existing code in the following ways:
    - Remove **$(SRCROOT)/WorklightAPI/include** from the header search path.
    - Remove **$(PROJECTDIR)/WorklightAPI/frameworks** from the frameworks search path.
    - Remove any references to the static **librarylibWorklightStaticLibProjectNative.a**.
4. In the **Build Phases** tab, remove the links to the following frameworks and libraries (these are re-added automatically by CocoaPods):
    - libWorklightStaticLibProjectNative.a
    - SystemConfiguration.framework
    - MobileCoreServices.framework
    - CoreData.framework
    - CoreLocation.framework
    - Security.framework
    - sqlcipher.framework
    - libstdc++.6.dylib
    - libz.dylib
5. Close Xcode.
6. Get the IBM MobileFirst Platform Foundation iOS SDK from CocoaPods. To get the SDK, complete the following steps:
    - Open **Terminal** at the location of your new Xcode project.
    - Run the `pod init` command to create a **Podfile** file.
    - Open the Podfile file that is in the root of the project with a text editor.
    - Comment out or remove the existing content.
    - Add the following lines and save the changes, including the iOS version:

      ```
      use_frameworks!
      platform :ios, 9.0
      pod 'IBMMobileFirstPlatformFoundation'
      ```

    - Specify additional pods in the file from the list above, if your app needs to use the additional functionality that they provide. For example, if your app uses OpenSSL, the **Podfile** might look like this:

      ```
      use_frameworks!
      platform :ios, 9.0
      pod 'IBMMobileFirstPlatformFoundation'
      pod 'IBMMobileFirstPlatformFoundationOpenSSLUtils'
      ```

> **Note:** The previous syntax imports the latest version of the **IBMMobileFirstPlatformFoundation** pod. If you are not using the latest version of MobileFirst, you need to add the full version number, including the major, minor, and patch numbers. The patch number is in the format YYYYMMDDHH. For example, for importing the specific patch version 8.0.2016021411 of the **IBMMobileFirstPlatformFoundation** pod the line would look like this:

```
pod 'IBMMobileFirstPlatformFoundation', '8.0.2016021411'
```

Or to get the last patch for the minor version number the syntax such is

```
pod 'IBMMobileFirstPlatformFoundation', '~>8.0.0'
```

- Verify that the Xcode project is closed.
- Run the `pod install` command.

This command installs the MobileFirst SDK **IBMMobileFirstPlatformFoundation.framework** and any other frameworks that are specified in the Podfile and their dependencies. It then generates the pods project, and integrates the client project with the MobileFirst SDK.

7. Open your **ProjectName.xcworkspace** file in Xcode by typing open **ProjectName.xcworkspace** from a command line. This file is in the same directory as the **ProjectName.xcodeproj** file.
8. Replace all of the existing MobileFirst imports of headers with a single entry of the following new umbrella header:

   Objective-C

   ```
   #import <IBMMobileFirstPlatformFoundation/IBMMobileFirstPlatformFoundation.h>
   ```

   Swift

   ```
   import IBMMobileFirstPlatformFoundation
   ```

   If you are using Push or JSONStore, you need to include an independent import.

   Push

   Objective-C

   ```
   #import <IBMMobileFirstPlatformFoundationPush/IBMMobileFirstPlatformFoundationPush.h>
   ```

   Swift

   ```
   import IBMMobileFirstPlatformFoundationPush
   ```

   JSONStore

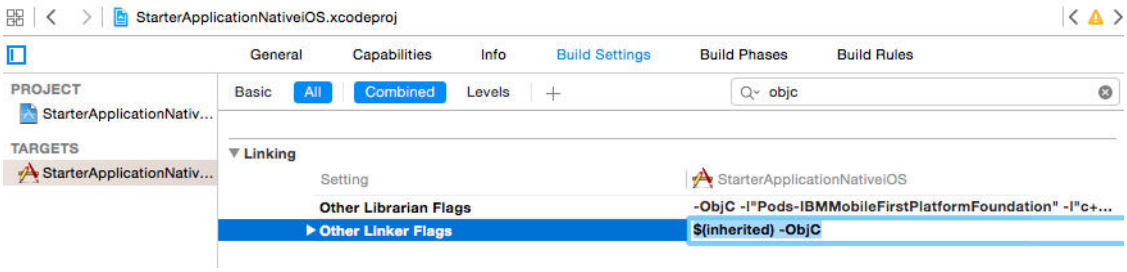
   Objective-C

   ```
   #import <IBMMobileFirstPlatformFoundationJSONStore/IBMMobileFirstPlatformFoundationJSONStore.h>
   ```

   Swift

   ```
   import IBMMobileFirstPlatformFoundationJSONStore
   ```

9. In the **Build Settings** tab, under **Other Linker Flags**, add `$(inherited)` at the beginning of the `-ObjC` flag. For example:



10. Beginning with Xcode 7, TLS must be enforced, see Enforcing TLS-secure connections in iOS apps.

Your application is now upgraded to work with the IBM MobileFirst Platform Foundation, v8.0 iOS SDK.

What to do next

Replace the client-side APIs that are discontinued or not in v8.0.

Migrating encryption in iOS

If your iOS application used OpenSSL encryption, you might want to migrate your app to the new v8.0 native encryption. Also, if you want to continue using OpenSSL, you must install some additional frameworks.

For more information on the iOS encryption options for migration, see Enabling OpenSSL for iOS (../../../application-development/sdk/ios/additional-information/#enabling-openssl-for-ios).

Updating the iOS code

After updating the iOS framework and making necessary configuration changes, a number of issues can be relevant to your specific application code.
The iOS API changes are listed in the table below.

API element	Migration path
• `[WLClient getWLDevice][WLClient transmitEvent:]` • `[WLClient setEventTransmissionPolicy]` • `[WLClient purgeEventTransmissionBuffer]`	Geolocation removed. Use native iOS or third-party packages for GeoLocation.
• `WL.Client.getUserInfo(realm, key)` • `WL.Client.updateUserInfo(options)`	No replacement.
`WL.Client.deleteUserPref(key, options)`	No replacement. You can use an adapter and the `MFP.Server.getAuthenticatedUser` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. server/html/MFP.Server.html?view=kc#MFP.Server.getAuthenticatedUser:) API to manage
`[WLClient getRequiredAccessTokenScopeFromStatus]`	Use `WLAuthorizationManager obtainAccessTokenForScope` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/WLAuthorizationManager.html?view=kc#//api/name/obtainAccessTokenF
`[WLClient login:withDelegate:]`	Use `WLAuthorizationManager login` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/WLAuthorizationManager.html?view=kc#//api/name/login:withCredentials:
`[WLClient logout:withDelegate:]`	Use `WLAuthorizationManager logout` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/WLAuthorizationManager.html?view=kc#//api/name/logout:withCompletio
• `[WLClient lastAccessToken]` • `[WLClient lastAccessTokenForScope:]`	Use `WLAuthorizationManager obtainAccessTokenForScope` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/WLAuthorizationManager.html?view=kc#//api/name/obtainAccessTokenF
• `[WLClient obtainAccessTokenForScope:withDelegate:]` • `[WLClient getRequiredAccessTokenScopeFromStatus:authenticationHeader:]`	Use `WLAuthorizationManager obtainAccessTokenForScope` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/WLAuthorizationManager.html?view=kc#//api/name/obtainAccessTokenF
`[WLClient isSubscribedToAdapter:(NSString *) adapteреventSource: (NSString *) eventSource`	Use Objective-C client-side push API for iOS apps (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. view=kc#nativeobjective-capiforandroidapps) from the IBMMobileFirstPlatformFoundationP
`[WLClient - (int) getEventSourceIDFromUserInfo: (NSDictionary *) userInfo]`	Use Objective-C client-side push API for iOS apps (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. view=kc#nativeobjective-capiforandroidapps) from the IBMMobileFirstPlatformFoundationP
`[WLClient invokeProcedure: (WLProcedureInvocationData *) ]`	Deprecated. Use `WLResourceRequest` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/WLResourceRequest.html?view=kc#/api/name/sendWithDelegate:) instea
`[WLClient sendUrlRequest:delegate:]`	Use `[WLResourceRequest sendWithDelegate:delegate]` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/WLResourceRequest.html?view=kc#/api/name/sendWithDelegate:) instea
`[WLClient (void) logActivity:(NSString *) activityType]`	Removed. Use an Objective C logger.
• `[WLSimpleDataSharing setSharedToken: myName value: myValue]` • `[WLSimpleDataSharing getSharedToken: myName]]` • `[WLSimpleDataSharing clearSharedToken: myName]`	Use the OS APIs to share tokens across applications.
`BaseChallengeHandler.submitFailure(WLResponse *)challenge`	Use `BaseChallengeHandler.cancel()` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/BaseChallengeHandler.html?view=kc).
`BaseProvisioningChallengeHandler`	No replacement. Device provisioning is now handled automatically by the security framewo
`ChallengeHandler`	For custom gateway challenges, use `GatewayChallengeHandler` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/SecurityCheckChallengeHandler.html?view=kc). For MobileFirst security-c `SecurityCheckChallengeHandler` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/SecurityCheckChallengeHandler.html?view=kc).
`WLChallengeHandler`	Use `SecurityCheckChallengeHandler` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/SecurityCheckChallengeHandler.html?view=kc).
`ChallengeHandler.isCustomResponse()`	Use `GatewayChallengeHandler.canHandleResponse()` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/GatewayChallengeHandler.html?view=kc).
`ChallengeHandler.submitAdapterAuthentication`	Implement similar logic in your challenge handler. For custom gateway challenge handlers, (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/GatewayChallengeHandler.html?view=kc). For MobileFirst security-check `SecurityCheckChallengeHandler` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref. ios/html/Classes/SecurityCheckChallengeHandler.html?view=kc).