Adding the MobileFirst Platform Foundation SDK to Windows 10 UWP Applications

Overview

To serve a native Windows 8 Universal application, MobileFirst Server must be aware of it. For this purpose, IBM MobileFirst Platform Foundation provides a Native API library, which contains a set of APIs and configuration files.

This tutorial explains how to generate the Windows 8 Universal Native API and how to integrate it with a native Windows Universal application. These steps are necessary for you to be able to use it later on for tasks such as connecting to MobileFirst Server, invoking adapter procedures, implementing authentication methods, and so on.

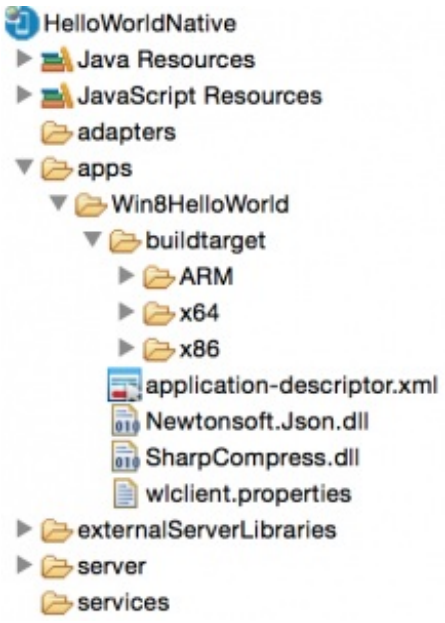
Prerequisite: Developers are expected to be proficient with Microsoft developer tools.

Creating and deploying a MobileFirst native API

CLI

1. Using the CLI (../../advanced-client-side-development/using-cli-create-build-manage-project-artifacts/), create a new MobileFirst project: `$ mfp create HelloWorldNative`
2. Go to the newly created project directory: `$ cd HelloWorldNative/`
3. Add a new Windows Universal native API: `$ mfp add api Win8HelloWorld -e windows8`
4. Navigate into the native API folder and run the command: `$ mfp push`. **Note:** This action is required for MobileFirst Server to recognize the application if it attempts to connect.

Studio



(https://developer.ibm.com/mobilefirstplatform/wp-content/uploads/sites/32/2015/04/Windows8UniversalProject.png)

1. In MobileFirst Studio, create a MobileFirst project and add a MobileFirst Native API.
2. In the **New MobileFirst Native API** dialog, enter your application name and select **Windows Universal** for the **Environment** field.
3. Right-click the generated NativeAPI folder (located in *your-projects*/apps/*your-nativeapi-app-name*) and select **Run As > Deploy Native API**.
   Note: This action is required in order for MobileFirst Server to recognize the application if it attempts to connect.

The MobileFirst native API contains several components:

- `worklight-windows8.dll` is a MobileFirst API library that you must copy to your native Windows 8 Universal project. This is contained within the "buildtarget" folder , under the respective hardware architecture.
- `Newtonsoft.Json.dll` is a library that provides JSON support.
- `SharpCompress.dll` is a library that provides compression support.
- `application-descriptor.xml` defines application metadata and security settings that MobileFirst Server enforces.
- `wlclient.properties` contains connectivity settings that a native Windows Universal application uses. You must copy this file to your native Windows Universal project.
- As with any MobileFirst project, you create the server configuration by modifying the files that are in the `server\conf` folder.

wlclient.properties

You can edit the *wlclient.properties* file to set connectivity information.

- wlServerProtocol – The communication protocol to MobileFirst Server, which is either http or https.
- wlServerHost – The host name of the MobileFirst Server instance.
- wlServerPort – The port of the MobileFirst Server instance.
- wlServerContext – The context root path of the application on MobileFirst Server.
- wlAppId – The application ID as defined in the `application-descriptor.xml` file.
- wlAppVersion – The application version.
- wlEnvironment – The target environment of the native application.
- wlPlatformVersion – The MobileFirst Studio version.
- languagePreferences – The list of preferred locales.

Creating and configuring a Windows Universal native application

1. Create a Windows Universal Application project or use an existing one.
2. Add as a *reference* `worklight-windows8.dll`, `Newtonsoft.Json.dll` and `SharpCompress.dll` files.Choose the right `worklight-windowsphone8.dll` from the folder that matches the architecture of the target device (ARM/x64/x86).
3. Copy the `wlclient.properties` file to the root of the native project.
4. In Visual Studio, open the **Properties** window of the `wlclient.properties` file and set the **Copy to**

Output Directory option to **Copy always**.

5. Add the following capabilities to the `Package.appxmanifest`:

   Internet (Client & Server)

   Private Networks (Client & Server)

> For more information, see the topic about developing native C# applications for Windows Universal, in the user documentation.

Tutorials to follow next

Now that your application contains the Native API library, you can follow the tutorials in the Native Windows 8 Development (../../native/windows8/) section to learn more about authentication and security, server-side development, advanced client-side development, notifications and more.