Adding the MobileFirst Platform Foundation SDK to Windows 8.1 Universal Applications

Overview

The MobileFirst Platform Foundation SDK provides a set of API methods enabling a developer to implement various MobileFirst features, such as: authentication and security mechanisms, notifications, resource requests, collecting analytics data and more.

> For a complete list of MobileFirst SDK abilities visit the user documentation (http://www-01.ibm.com/support/knowledgecenter/SSHS8R_8.0.0/wl_welcome.html).

In this tutorial you will learn how to add the MobileFirst Native SDK using Nuget to either a new or existing Windows 8.1 Universal application. You will also learn how to configure the MobileFirst Server to recognize the application, as well as find information about the MobileFirst configuration files that are added to the project.

Pre-requisites:

- Microsoft Visual Studio 2013 or 2015 and MobileFirst Developer CLI installed on the developer workstation.
- *Optional* MobileFirst Server to run a locally.
- Make sure you have read the Setting up your MobileFirst development environment (../../setting-up-the-mobilefirst-development-environment) tutorial.

Jump to:

Adding the MobileFirst Native SDK

Follow the below instructions to manually add the MobileFirst Native SDK to either a new or existing Visual Studio project, and registering the application in the MobileFirst Server.

Before starting, make sure the MobileFirst Server is running.
If using a locally installed server: From a **Command-line** window, navigate to the server's **scripts** folder and run the command: `./start.sh` in Mac and Linux or `start.cmd` in Windows.

Creating and registering the application

1. Create a Windows 8.1 Universal project using Visual Studio 2013/2015 or use an existing project.
2. Open the **Command-line** and navigate to the root of the Visual Studio project.
3. Run the command:

```
mfpdev app register
```

The `mfpdev app register` CLI command first connects to the MobileFirst Server to register the application, followed by generating the `mfpclient.resw` file at the root of the Visual Studio project, and adding to it the metadata that identifies the MobileFirst Server.

> **Tip:** The application registration can also be performed from the MobileFirst Operations Console:
> 1. Open your browser of choice and load the MobileFirst Operations Console using the address `http://localhost:9080/mfpconsole/`. You can also open the console from the **Command-line** using the CLI command `mfpdev server console`.
> 2. Click on the "Create new" button next to "Applications" to create a new application and follow the on-screen instructions.
> 3. After successfully registering your application you can optionally download a "skeleton" Visual Studio project pre-bundled with the MobileFirst Native SDK.

4. Run the command:

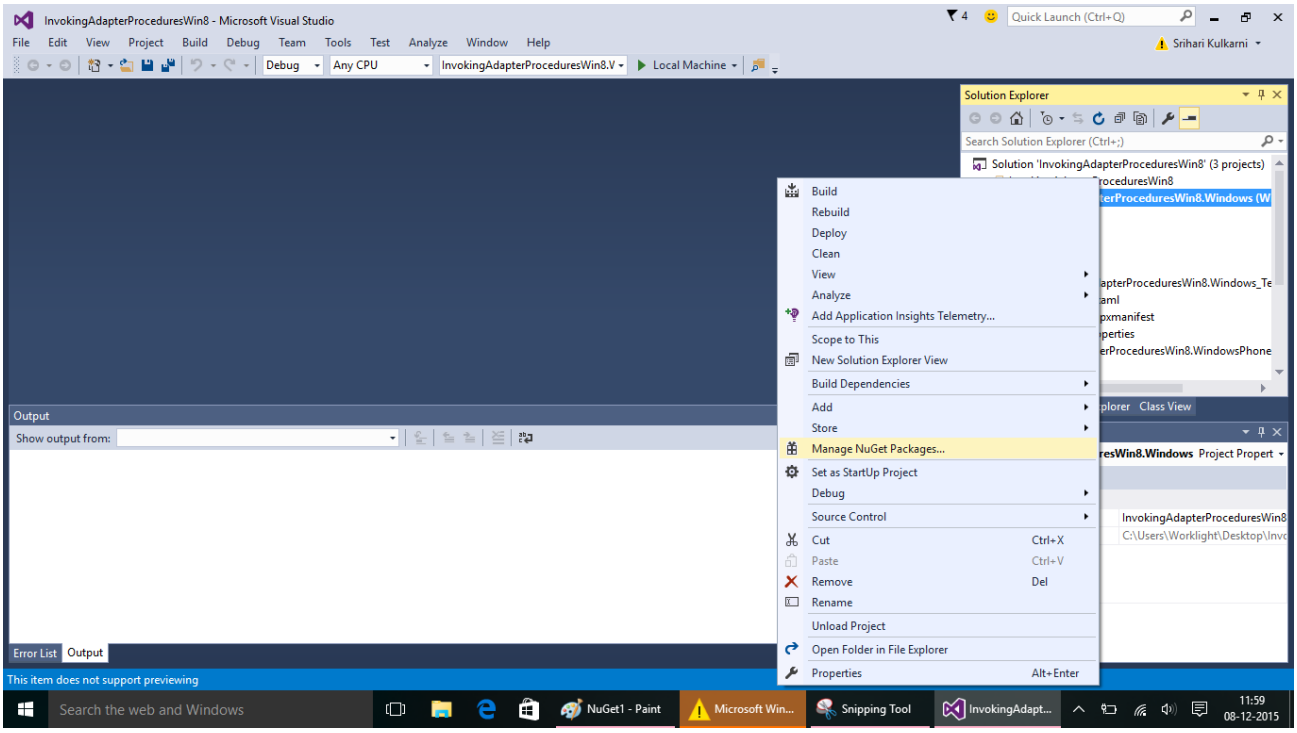
```
mfpdev app pull
```

The `mfpdev app pull` CLI command creates the **mobilefirst** folder at the root of the Visual Studio project and downloads into it the `application-descriptor.json` file, containing application configuration data.
These files are further explained in the Generated MobileFirst Native SDK artifacts section below.

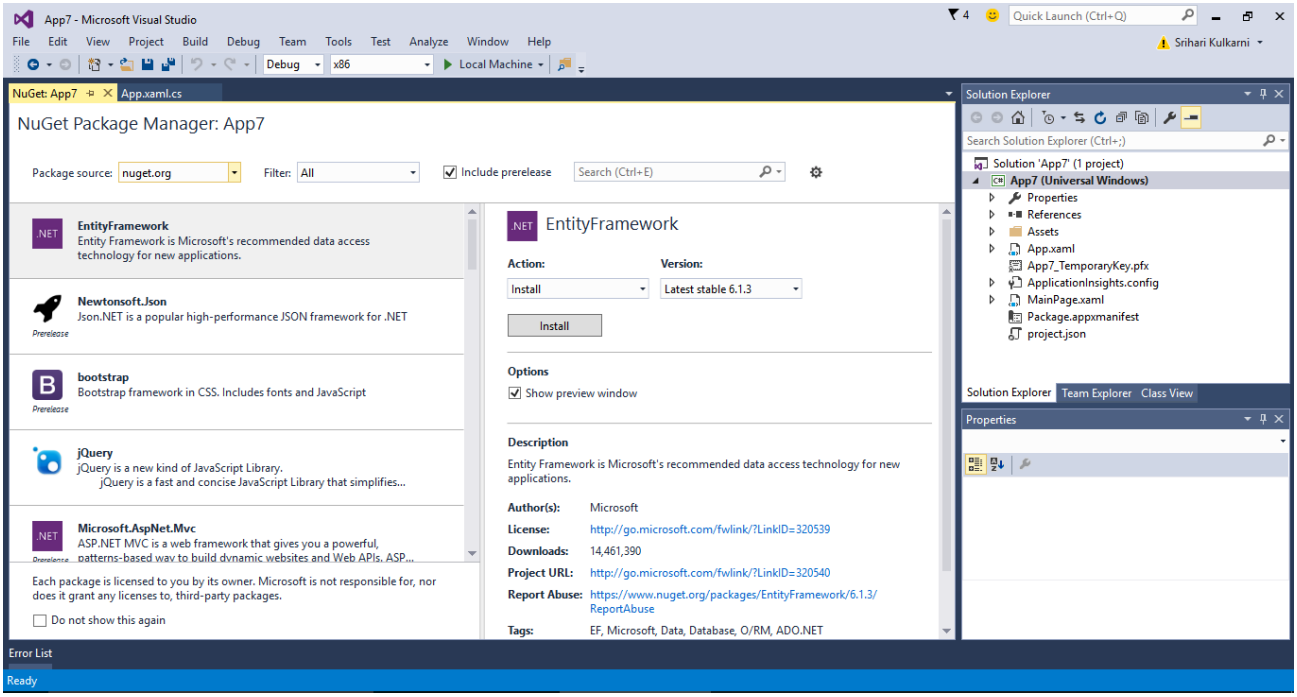
> **Tip:** Learn more about the various CLI commands in the Using MobileFirst Developer CLI to manage MobileFirst artifacts (../../client-side-development/using-mobilefirst-developer-cli-to-manage-mobilefirst-artifacts/) tutorial.

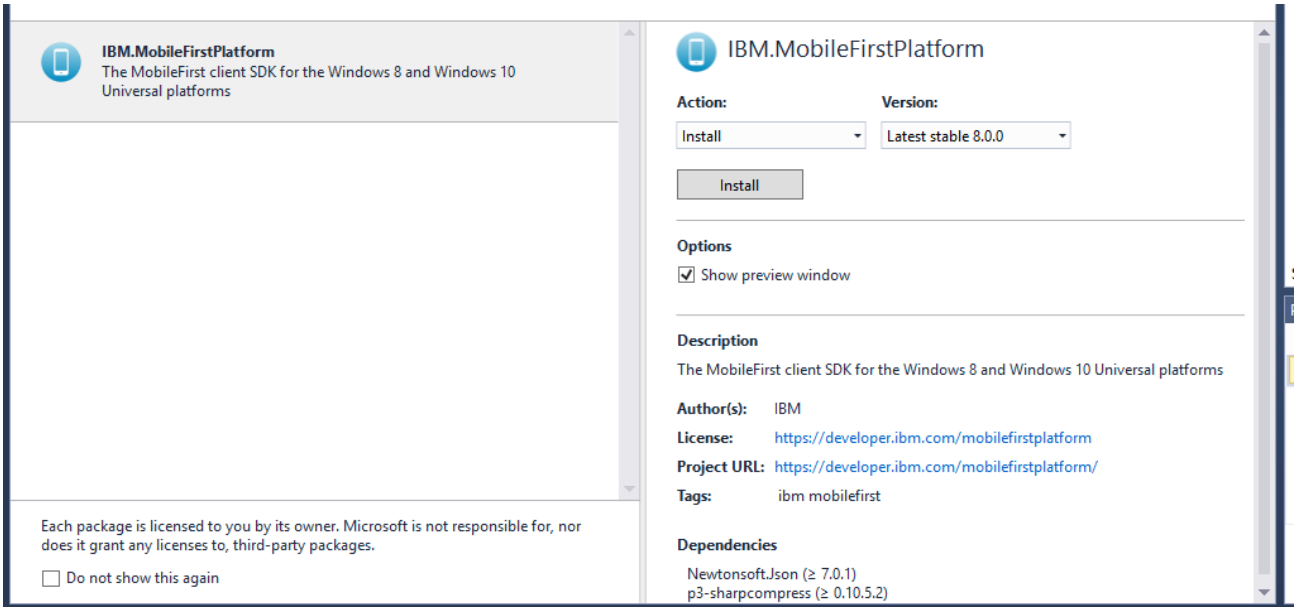
Adding the SDK

1. To import worklight studio packages, NuGet package manager is used. NuGet is the package manager for the Microsoft development platform including .NET. The NuGet client tools provide the ability to produce and consume packages. The NuGet Gallery is the central package repository used by all package authors and consumers.
2. Open the Windows 8.1 Universal project in Visual studio 2013/2015. Right-click the project solution and select **Manage Nuget packages**.



3. In the search option, search for "IBM MobileFirst Platform". Choose **IBM.MobileFirstPlatform.8.0.0.0**.





4. Click **Install**. This installs the IBM MobileFirst Platform Native SDK and its dependencies.

Generated MobileFirst Native SDK artifacts

Two MobileFirst-related artifacts are available in the Android Studio project after it has been integrated with the MobileFirst Native SDK: the `mfpclient.resw` and the `application-descriptor.json` file.

mfpclient.resw

Located at the root of the project, this file contains server connectivity properties and is user-editable:

- `protocol` – The communication protocol to MobileFirst Server. Either `HTTP` or `HTPS`.
- `host` – The hostname of the MobileFirst Server instance.
- `port` – The port of the MobileFirst Server instance.
- `wlServerContext` – The context root path of the application on the MobileFirst Server instance.
- `languagePreference` - Sets the default language for client sdk system messages

In Visual Studio, open the **Properties** window of the `mfpclient.resw` file and set the **Copy to Output Directory** option to **Copy always**.

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

- Internet (Client & Server)
- Private Networks (Client & Server)

application-descriptor.json

Located in the **<visual-studio-project-root-directory>/mobilefirst** folder, this file contains application configuration settings such as its `bundleId` and `version` and is user-editable.

The file can be edited either locally or via the MobileFirst Operations Console.
If edited locally, the MobileFirst Server can be updated by running the CLI command: `mfpdev app push`.
The file can also be updated by pulling from the server its latest revision by running the CLI command: `mfpdev app pull`.

```
{
   "applicationKey": {
      "packageName": "com.samplePackage",
      "version": "1.0",
      "clientPlatform":"visualstudio"
   }

   ...
   ...
   ...
}
```

Tutorials to follow next

Now that your application contains the Native API library, you can follow the tutorials in the Native Windows 8.1 Development (../../native/windows8/) section to learn more about authentication and security, server-side development, advanced client-side development, notifications and more.