Adding the MobileFirst Platform Foundation SDK to Android 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 Gradle and Maven Central to either a new or existing Android 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: Android Studio and MobileFirst CLI installed on the developer workstation.
Make sure you have read the Setting up your MobileFirst development environment (../../setting-up-the-mobilefirst-development-environment) tutorial.

Jump to:

Manually Adding the MobileFirst Native SDK

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

Before starting, make sure the MobileFirst Server is running.
From a **Terminal** window run the command:

```
mfpdev server start
```

1. Create an Android project or use an existing one.
2. Open **Terminal** and navigate to the root of the Android 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.properties` file in the **./app/src/main/assets/** folder of the Android 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 **Terminal** 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" Android 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 Android 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 CLI to manage MobileFirst artifacts (../../client-side-development/using-cli-to-manage-mobilefirst-artifacts/) tutorial.

5. In **Project > Gradle scripts**, select **build.gradle (Module: app)**.
6. Add the following lines below `apply plugin: 'com.android.application'`:

```
repositories{
    //jcenter() ---- remove below maven line and bring back jcenter line o
nce we go live with the tutorials
    maven {
        url "http://ibobs-Mac-mini.local:8081/nexus/content/repositories/s
napshots/"
    }
}
```

7. Add the following inside `android`:

```
packagingOptions {
    pickFirst 'META-INF/ASL2.0'
    pickFirst 'META-INF/LICENSE'
    pickFirst 'META-INF/NOTICE'
}
```

8. Add the following lines inside `dependencies`:

```
compile group: 'com.ibm.mobile.foundation',
name: 'ibmmobilefirstplatformfoundation',
version: '8.0.Beta1-SNAPSHOT',
ext: 'aar',
transitive: true
```

Or:

```
```xml
compile 'com.ibm.mobile.foundation:ibmmobilefirstplatformfoundation:8.0.Beta1-S
NAPSHOT'
```
```

1. Add the following permissions to the `AndroidManifest.xml` file:

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

2. Add the MobileFirst UI activity:

```
<activity android:name="com.worklight.wlclient.ui.UIActivity" />
```

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.properties` and the `application-descriptor.json` file.

mfpclient.properties

Located at the **./app/src/main/assets/** folder of the Android Studio 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

application-descriptor.json

Located in the **<android-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":"android"
    }

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

Tutorials to follow next

Now that the application is integrated with the MobileFirst Native SDK you can continue reading tutorials for Native Android development (../../android-tutorials/) to learn more about authentication and security, server-side development, notifications, and more.