

# Android end-to-end demonstration

fork and edit tutorial (<https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/8.0/quick-start/android/index.md>)

| report issue (<https://github.ibm.com/MFPSamples/DevCenter/issues/new>)

## Overview

The purpose of this demonstration is to experience an end-to-end flow where an application and an adapter are registered using the MobileFirst Operations Console, an "skeleton" Android Studio project is downloaded and edited to call the adapter, and the result is printed to the log - verifying a successful connection with the MobileFirst Server.

### Prerequisites:

- Android Studio
- MobileFirst Developer CLI (download  
([file:///home/travis/build/MFPSamples/DevCenter/\\_site/downloads](file:///home/travis/build/MFPSamples/DevCenter/_site/downloads)))
- *Optional*. Stand-alone MobileFirst Server (download  
([file:///home/travis/build/MFPSamples/DevCenter/\\_site/downloads](file:///home/travis/build/MFPSamples/DevCenter/_site/downloads)))

## 1. Starting the MobileFirst Server

If a remote server was already set-up, skip this step.

From a **Command-line** window, navigate to the server's folder and run the command: `./run.sh` in Mac and Linux or `run.cmd` in Windows.

## 2. Creating an application

In a browser window, open the MobileFirst Operations Console by loading the URL: `http://your-server-host:server-port/mfpconsole`. If running locally, use: `http://localhost:9080/mfpconsole` (`http://localhost:9080/mfpconsole`). The username/password are `admin/admin`.

1. Click on the "New" button next to **Applications**
  - Select the **Android** platform
  - Enter **com.ibm.mfp.sample.mfpstarterandroid** as the **application identifier**
  - Enter **1.0** as the **version** value



2. Click on the **Get Starter Code** tile and select to download the Android mobile app scaffold.



### 3. Editing application logic

1. Open the Android Studio project and import the project.
2. Select the **app/java/com/mfp/sample/MainActivity.java** file and:

- Add the following imports:

```
import java.net.URI;
import android.util.Log;
```

- Paste the following code snippet, inside the `protected void onCreate()` function:

```

WLClient.createInstance(this);
URI adapterPath = null;
try {
    adapterPath = new URI("/adapters/javaAdapter/users/world");
} catch (URISyntaxException e) {
    e.printStackTrace();
}

WLResourceRequest request = new WLResourceRequest(adapterPath, WLResourceRequest.GET);
request.send(new WLResponseListener() {
    @Override
    public void onSuccess(WLResponse wlResponse) {
        // Will print "Hello world" in LogCat.
        Log.i("MobileFirst Quick Start", "Success: " + wlResponse.getResponseText());
    }

    @Override
    public void onFailure(WLFailResponse wlFailResponse) {
        Log.i("MobileFirst Quick Start", "Failure: " + wlFailResponse.getErrorMsg());
    }
});

```

## 4. Creating an adapter

Click on the "New" button next to **Adapters**.

Alternatively, download this prepared .adapter artifact (../javaAdapter.adapter) and deploy it from the MobileFirst Operations Console using the **Actions → Deploy adapter** action.

1. Select the **Actions → Download sample** option. Download the "Hello World" **Java** adapter sample.

If Maven and MobileFirst Developer CLI are not installed, follow the on-screen **Set up your development environment** instructions.

2. From a **Command-line** window, navigate to the adapter's Maven project root folder and run the command:

```
mfpdev adapter build
```

3. When the build finishes, deploy it from the MobileFirst Operations Console using the **Actions → Deploy adapter** action. The adapter can be found in the **[adapter]/target** folder.



