

# iOS end-to-end demonstration

fork and edit tutorial (<https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/8.0/quick-start/ios/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" Xcode 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:

- Xcode
- 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`.

## 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 **iOS** platform
  - Enter **com.ibm.mfp.MFPStarterIOSObjectiveC** or **com.ibm.mfp.MFPStarterIOSSwift** as the **application identifier** (depending on which mobile app scaffold you will download next)
  - Enter **1.0** as the **version** value

The screenshot shows the MobileFirst Operations Console interface. The left sidebar contains a navigation menu with 'Dashboard', 'Runtimes', 'Applications' (highlighted), 'Adapters', 'Settings', 'Devices', and 'Error Log'. The 'Applications' section has a 'New' button. The main content area is titled 'Register an Application' and contains the following fields and options:

- Application Name**: A text input field.
- Optional display name of the Application**: A text input field.
- Choose Platform \***: Radio buttons for 'Android', 'iOS' (selected), and 'Windows'.
- Bundle ID \***: A text input field with the placeholder 'Application Identifier; case sensitive'.
- Version \***: A text input field with the placeholder 'Application Version'.
- Register application**: A blue button at the bottom.

The top right of the console shows 'Analytics Console', 'Hello, admin', and an information icon.

2. Click on the **Get Starter Code** tile and select to download the iOS Objective-C or Swift mobile app scaffold.



### 3. Editing application logic

1. Open the Xcode project project by double-clicking the **.xcworkspace** file.
2. Select the **[project-root]/ViewController.m/swift** file and paste the following code snippet, replacing the existing `viewDidLoad()` function:

In Objective-C:

```
- (void)viewDidLoad {
    [super viewDidLoad];

    NSURL* url = [NSURL URLWithString:@"~/adapters/javaAdapter/users/world"];
    WLResourceRequest* request = [WLResourceRequest requestWithURL:url method:WLHttpMethodGet];

    [request sendWithCompletionHandler:^(WLResponse *response, NSError *error) {
        if (error != nil){
            NSLog(@"Failure: %@",error.description);
        }
        else if (response != nil){
            // Will print "Hello world" in the Xcode Console.
            NSLog(@"Success: %@",response.responseText);
        }
    }];
}
```

In Swift:

```
override func viewDidLoad() {
    super.viewDidLoad()

    let url = NSURL(string: "~/adapters/javaAdapter/users/world")
    let request = WLResourceRequest(URL: url, method: WLHttpMethodGet)

    request.sendWithCompletionHandler { (WLResponse response, NSError error) -> Void in
        if (error != nil){
            NSLog("Failure: " + error.description)
        }
        else if (response != nil){
            NSLog("Success: " + response.responseText)
        }
    }
}
```

### 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.

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

- mfpdev adapter build

- MobileFirst Operations Console

Analytics Console

Hello, admin

Dashboard

Runtime

mfpmfp

ApplicationsNew

MFPStarterIOSSwift

AdaptersNew

No adapter deployed

Settings

Devices

Error Log

Home > mfp > Create a new Adapter

Create a new Adapter

Deploy Adapter

Adapters are used to securely connect back-end systems to client applications and cloud services. Adapters are built as Maven projects and can be written in JavaScript or Java.

Follow these steps to create an adapter.[Hide guide](#)

1Set up your development environment.

2Create

There are three ways to start developing adapter projects:

Console

CLI

Maven

Start with one of the packaged [sample projects](#)

3Develop

4Build

5Deploy

1. In Xcode, select the **mfpclient.plist** file and edit the **host** property with the IP address of the MobileFirst Server.
2. Press the **Play** button.

- Clicking on the **Test Server Connection** button will display **Obtained Access Token Successfully**.
- If the application was able to connect to the MobileFirst Server, a resource request call using the Java adapter will take place.

```
MyApplication
Date = "Tue, 19 Jan 2016 06:14:40 GMT";
"Transfer-Encoding" = Identity;
"X-Powered-By" = "Servlet/3.1.1";
}
Response Data:
{"access_token": "eyJhbGciOiJIUzI1NiIsInp1IjE7eYyIjo1VFQ6b0IsIn4iOiJBTTEZBZD40RWRkMktgeWdWMN3I4cUNmZEUtM0kyazS0NXpnNmREZF9xczhmdm5SZ2xmPvcvR2VrYjRfMnQ0T0dHOENWNUNlNDQFTkdjXDI3MDNDEwdlJWm5S2hhvbmVtLWY0YTU9ISXFvZSl5SkEwdVpdzjY5GhVWjIXXkVSI2VGVlZj09ZCZlF0LWLR5SztZnoIZnXwLVmFMFVzdihU893QKjsMVucUl3VKR3TL2ZJKTudsMEVVCy1BaZmt0kkStFSUB01paSLuckSMelJXa01ITHzMTD0TV6b3MNVTExKNZZLQ8twadJCxG1TbtJTjNFuRhGN2DMRW95BUruVEqFK1Qw9mMuLS50ZNJHWVTRwczlTzQ03JOVLWS9h71YbxExKOuLDcczkZHGg38clMRWD0VzImFVNCKedRtbQhNMWz0dofFRKhJHwIwaz1Ki1oY2Jz0duKwM1tOT1Lngh0MBDZ1TKSzQMWEtZNGRLNWGU0T3in19_yEp3cKMl0J1bz0uwMtlmcIncIslnJY16tmNzTCcwQVYNWNOJPCfpJ2qlQwV15IemvXesLmpKc3JipFrnpFalIPzqW148Z4PB2644qVhdyKMSgdLT67J05t10GCLpZ0Sn0m_w3SPCVKw3SG0Q2yiAbBKbfFN5_6VEAMer4Y8rJ08ml1VCFSZgb3b2E84Z1VLWghY7YnGHdKjKKXj1Wmb3LI2DNdZdc_w_v4yk6pj7mXLrs6zxHGX84gyBMWGBa7TFnPLXdGdg","token_type": "Bearer","expires_in": 3599,"scope": ""}
Status code=200
2016-01-19 08:14:40.418 MyApplication[93738:36590517] [OCLogger printMessage:withDataAndLevelTag:andPackage:] [Line 1005] [DEBUG] [WL_AFHTTPSessionManagerWrapper_PACKAGE] ~-[WLAFHTTPSessionManagerWrapper start] in WLAFHTTPSessionManagerWrapper.m:372 : Starting the request with URL http://19808/mfp/api/adapters/javaAdapter/users/world
2016-01-19 08:14:40.440 MyApplication[93738:36590517] [OCLogger printMessage:withDataAndLevelTag:andPackage:] [Line 1005] [DEBUG] [WL_AFHTTPSessionManagerWrapper_PACKAGE] ~-[WLAFHTTPSessionManagerWrapper requestFinished:responseObject:] in WLAFHTTPSessionManagerWrapper.m:388 : Request Success
2016-01-19 08:14:40.440 MyApplication[93738:36590517] [OCLogger printMessage:withDataAndLevelTag:andPackage:] [Line 1005] [DEBUG] [WL_AFHTTPSessionManagerWrapper_PACKAGE] ~-[WLAFHTTPSessionManagerWrapper Adapter invocation response: Hello world
2016-01-19 08:14:40.441 MyApplication[93738:36590517] Adapter invocation response: Hello world
```

**Note:** Xcode 7 enables Application Transport Security (ATS)

([https://developer.apple.com/library/ios/releasenotes/General/WhatsNewIniOS/Articles/iOS9.html#//apple\\_ref/doc/uid/TP40016198-SW14](https://developer.apple.com/library/ios/releasenotes/General/WhatsNewIniOS/Articles/iOS9.html#//apple_ref/doc/uid/TP40016198-SW14)) by default.

To complete the tutorial, disable ATS (<http://iosdevtips.co/post/121756573323/ios-9-xcode-7-http-connect-server-error>).

1. In Xcode, right-click the **[project]/info.plist** file → **Open As** → **Source Code**
2. Paste the following:

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSAllowsArbitraryLoads</key>
  <true/>
</dict>
```

## Next steps

Learn more on using adapters in applications, and how to integrate additional services such as Push Notifications, using the MobileFirst security framework and more:

- Review the Using the MobileFirst Platform Foundation ([../using-the-mfpf-sdk/](#)) tutorials
- Review the Adapters development ([../adapters/](#)) tutorials
- Review the Authentication and security tutorials ([../authentication-and-security/](#))
- Review the Notifications tutorials ([../notifications/](#))
- Review All Tutorials ([../all-tutorials](#))