iOS end-to-end demonstration

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))
- *Optional*. Stand-alone MobileFirst Server (download (file:////home/travis/build/MFPSamples/DevCenter/_site/downloads))

1. Starting the MobileFirst Server

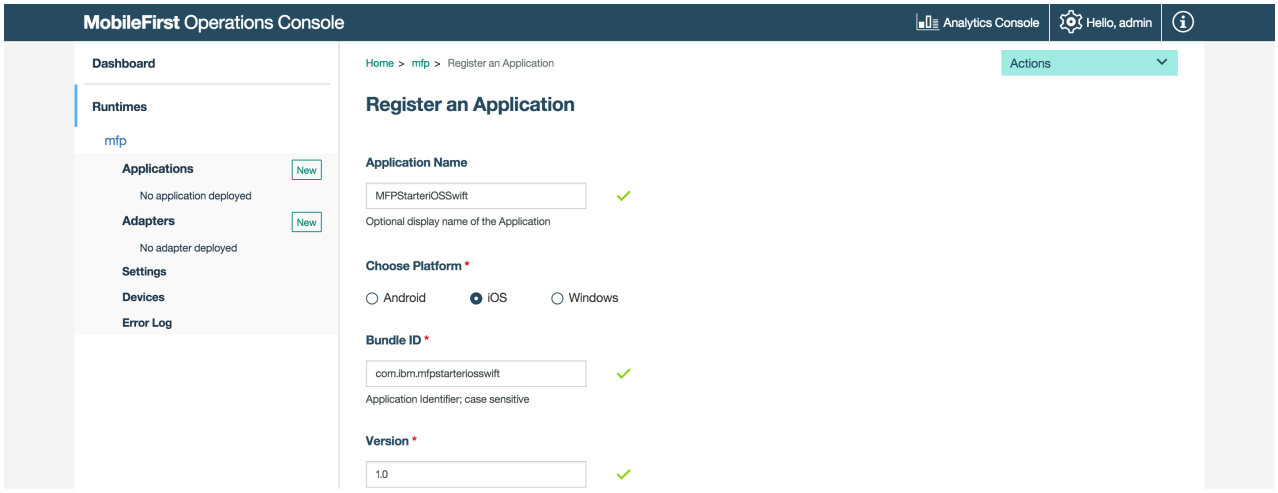
> If a remote server was already set-up, skip this step.

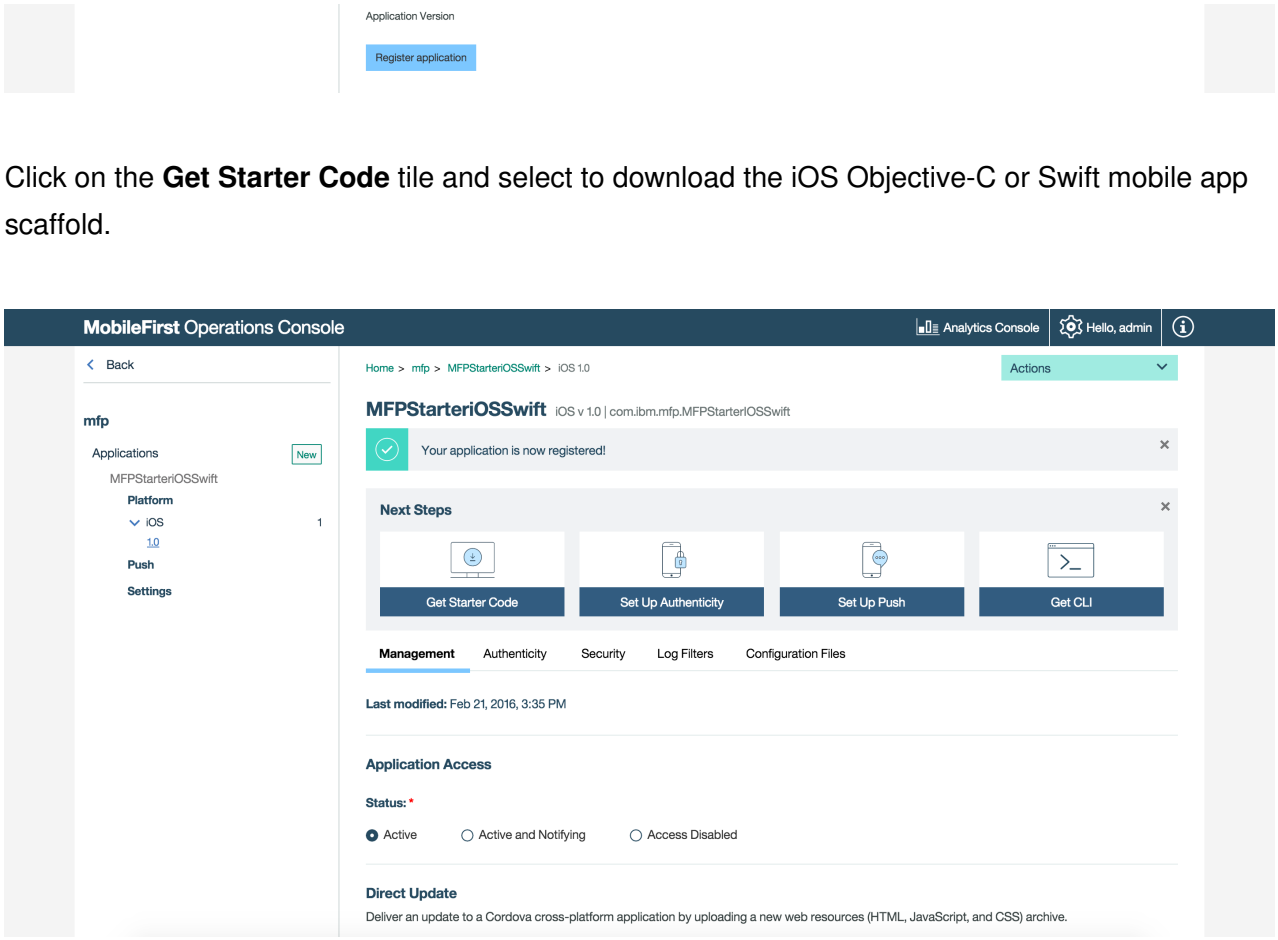
From a **Command-line** window, navigate to the server's folder and run the command: `./run.sh`.

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.mfpstarteriosobjectivec** or **com.ibm.mfpstarteriosswift** as the **application identifier** (depending on which mobile app scaffold you will download next)
    - Enter **1.0** as the **version** value
    - Click on **Register application**





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 `getAccessToken()` function:

   In Objective-C:

```
- (void)testServerConnection {
    _connectionStatusText.text = @"Connecting to Server...";
    [[WLAuthorizationManager sharedInstance] obtainAccessTokenForScope: @"" withCompletionH
andler:^(AccessToken *accessToken, NSError *error) {
        if (error != nil){
            NSLog(@"Failure: %@",error.description);
            _connectionStatusText.text = @"Client Failed to connect to Server";
        }
        else if (accessToken != nil){
            NSLog(@"Success: %@",accessToken.value);
            _connectionStatusText.text = @"Client has connected to Server";
            NSURL* url = [NSURL URLWithString:@"/adapters/javaAdapter/users/world"];
            WLResourceRequest* request = [WLResourceRequest requestWithURL:url method:WLHtt
pMethodGet];
            [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:

```
@IBAction func getAccessToken(sender: AnyObject) {
    connectionStatusWindow.text = "Connecting to Server...";
    print("Testing Server Connection")
    WLAuthorizationManager.sharedInstance().obtainAccessTokenForScope(nil) { (token, error) -
> Void in
        if (error != nil) {
            self.connectionStatusWindow.text = "Client Failed to connect to Server"
            print("Did not Recieved an Access Token from Server: " + error.description)
        } else {
            self.connectionStatusWindow.text = "Client has connected to Server"
            print("Recieved the Following Access Token value: " + token.value)
            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

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

Alternatively, click on the "New" button next to **Adapters**.

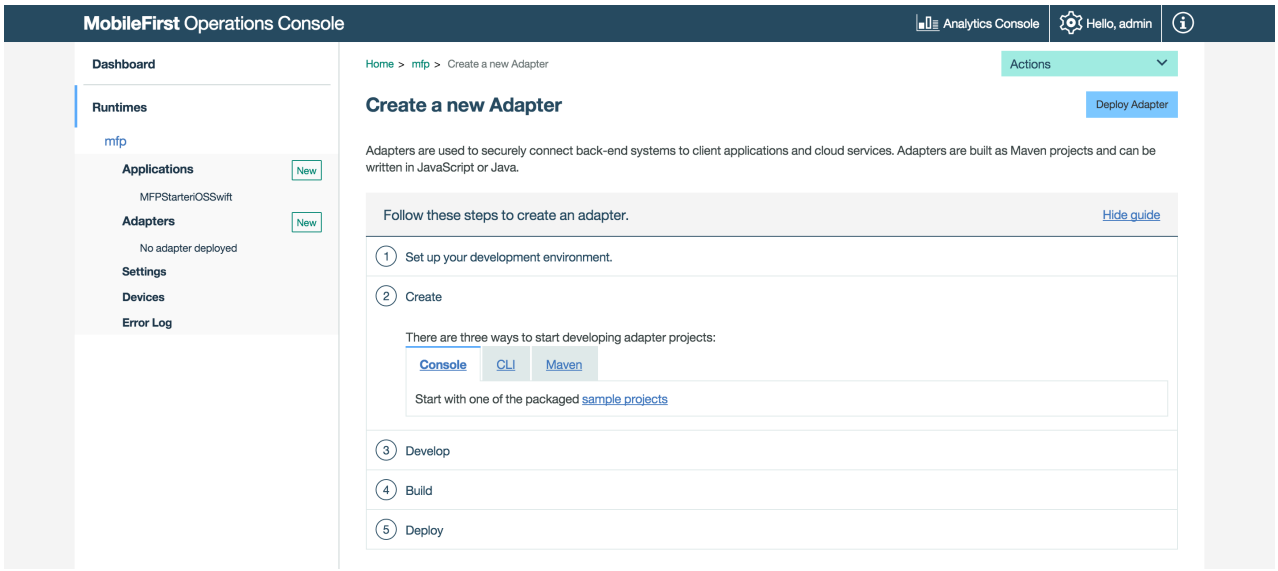
1. Select the **Actions → Download sample** option. Download the "Hello World" **Java** adapter sample.

   > 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.



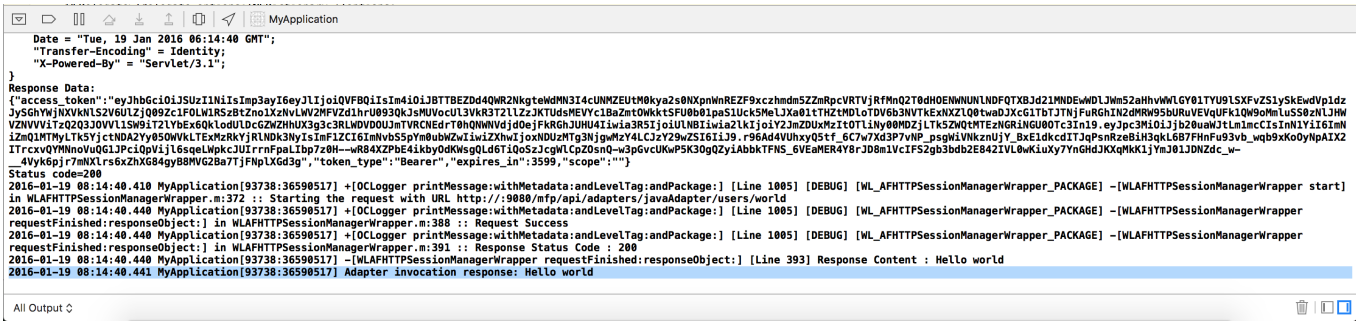
5. Testing the application

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.

Results

- Clicking on the **Test Server Connection** button will display **Client has connected to server**.
- If the application was able to connect to the MobileFirst Server, a resource request call using the Java adapter will take place.

The adapter response is then printed in the Xcode Console.



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)