Adding the MobileFirst Foundation SDK to Web Applications

Overview

In this tutorial you will learn how to register a web application with the MobileFirst Server, as well as downloading and adding the MobileFirst SDK to web applications.

The MobileFirst Web SDK is provided as a set of JavaScript files, and is available at NPM (https://www.npmjs.com/package/ibm-mfp-web-sdk).
The SDK is comprised of the following files:

- **ibmmfpf.js** - the core of the SDK.
- **ibmmfpfanalytics.js** - provides support for MobileFirst Foundation Analytics.

Prerequisite: to run NPM commands, Node.js (https://nodejs.org) is required.

Jump to:

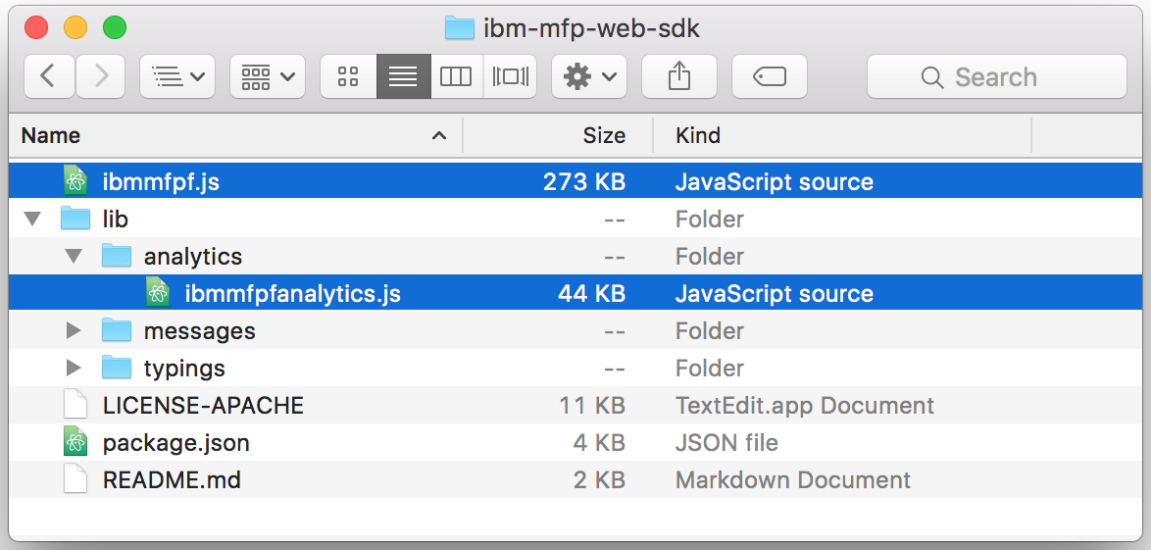
Adding the MobileFirst Web SDK

To add the SDK to new or existing web applications, first download it to your workstation and then add it to your web application.

Downloading the SDK

1. From a **command-line** window, navigate to your web application's root folder.
2. Run the command: `npm install ibm-mfp-web-sdk`.

This creates the following directory structure:



Adding the SDK

To add the MobileFirst Web SDK, reference it in standard fashion in the web application.
The SDK also supports AMD (https://en.wikipedia.org/wiki/Asynchronous_module_definition), so you can use Module Loaders such as RequireJS (http://requirejs.org/) to load the SDK.

Standard

Reference the **ibmmfpf.js** file in the `HEAD` element.

```
<head>
   ...
   ...
   <script type="text/javascript" src="node_modules/ibm-mfp-web-sdk/ibmmfpf.js"></script>
</head>
```

Using Require JS

HTML

```
<script type="text/javascript" src="node_modules/requirejs/require.js" data-main="index"></script>
```

JavaScript

```
require.config({
   'paths': {
      'mfp': 'node_modules/ibm-mfp-web-sdk/ibmmfpf'
   }
});

require(['mfp'], function(WL) {
   // application logic.
});
```

> ❗ **Important:** If adding Analytics support, place the **ibmmfpfanalytics.js** file reference **before** the **ibmmfpf.js** file reference.

Initializing the MobileFirst Web SDK

Initialize the MobileFirst Web SDK by specifying the **context root** and **application ID** values in the main JavaScript file of your web application:

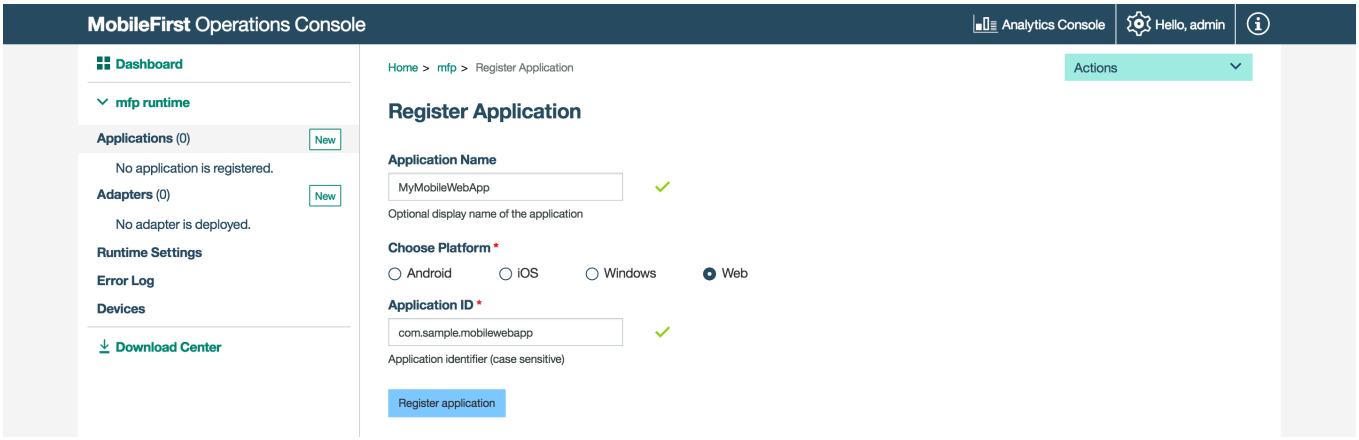
```
var wlInitOptions = {
    mfpContextRoot : '/mfp', // "mfp" is the default context root in the MobileFirst Development server
    applicationId : 'com.sample.mywebapp' // Replace with your own value.
};

WL.Client.init(wlInitOptions).then (
    function() {
        // Application logic.
});
```

- **mfpContextRoot:** the context root used by the MobileFirst Server.
- **applicationId:** the application package name, as defined while registering the application.

Registering the web application

1. Open your browser of choice and load the MobileFirst Operations Console using the address `http://localhost:9080/mfpconsole/`.
2. Click the "New" button next to "Applications" to create a new application.
3. Select **Web** as the platform, and provide its name and its application identifier. Then, click **Register application**.



Updating the MobileFirst Web SDK

SDK releases can be found in the SDK NPM repository (https://www.npmjs.com/package/ibm-mfp-web-sdk).
To update the MobileFirst Web SDK with the latest release:

1. Navigate to the root folder of the web application.
2. Run the command: `npm update ibm-mfp-web-sdk`.

Same-origin policy

Because web resources may be hosted on different a server machine than the one that MobileFirst Server is installed on, this may trigger a same-origin policy (https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy) violation.

Same-origin policy is a restriction embosed on web browsers. For example, if an application is hosted on the domain **example.com**, it is not allowed for the same application to also access contect that is available on another server, or for that matter, from the MobileFirst Server.

Web apps that are using the MobileFirst Web SDK should be handled in a supporting topology, for example by using a Reverse Proxy to internally redirect requests to the appropriate server while maintaining the same single origin.

Alternatives

The policy requirements can be satisfied by using either of the following methods:

- Serving the web application resources, for example, from the same WebSphere Liberty profile application server that is used in the MobileFirst Development Kit.
- Using Node.js as a proxy to redirect application requests to the MobileFirst Server.

> Learn more in Setting up the Web development environmnt (../../setting-up-your-development-environment/web-development-environment) tutorial

Secure-origins policy

When using Chrome during development, the browser might not allow an application to load if using both HTTP and a host that **is not** "localhost". This is due to the secure-origins policy implemented and used by default in this browser.

To overcome this, you can start the Chrome browser with the following flag:

```
--unsafely-treat-insecure-origin-as-secure="http://replace-with-ip-address-or-host:port-number" --user-data-dir=/test-to-new-user-profile/myprofile
```

- Replace "test-to-new-user-profile/myprofile" with the location of a folder that will act as a new Chrome user profile for the flag to work.

Read more about Secure Origins in this Chormium developer document (https://www.chromium.org/Home/chromium-security/prefer-secure-origins-for-powerful-new-features).

Tutorials to follow next

With the MobileFirst Web SDK now integrated, you can now:

- Review the Using the MobileFirst Foundation SDK tutorials (../../using-the-mfpf-sdk/)
- Review the Adapters development tutorials (../../adapters/)
- Review the Authentication and security tutorials (../../authentication-and-security/)
- Review All Tutorials (../../all-tutorials)