Using MobileFirst Developer CLI to Manage MobileFirst Artifacts

Overview

IBM MobileFirst Platform Foundation provides a Command Line Interface (CLI) tool to easily create, manage, push, and register MobileFirst client and server artifacts. The CLI tool enables you to use your preferred code editors or alternative IDEs.

By using the CLI, you can create and manage the following types of applications: Classic MobileFirst Hybrid applications, Cordova-based applications (iOS and Android) using the MobileFirst JavaScript SDK as a plug-in, and Native applications using the MobileFirst Native SDK (iOS, Android, Windows Universal, Windows Phone Silverlight).

You can also create, register, and manage MobileFirst adapters to either local or remote MobileFirst Server instances, and administer projects from the command line or via REST services, or from the MobileFirst Operations Console.

For more information regarding SDK integration in native applications, see the tutorials in the Configuring the MobileFirst Platform Foundation SDK category (../../configuring-the-mfpf-sdk).
For more information regarding SDK integration in Cordova applications, see the Integrating MobileFirst Platform Foundation SDK in Pure Cordova Applications (../integrating-mfpf-sdk-in-pure-cordova-applications/) tutorial.

> Learn more about the MobileFirst CLI in the "MobileFirst Platform Command Line Interface" topic in the user documentation

Agenda

- Installing the Command Line Interface
- Creating MobileFirst back-end projects
- Managing MobileFirst Server instances
- Creating hybrid applications and environments
- Adding MobileFirst Native SDK to native apps
- Creating and testing adapters
- Exporting and importing MobileFirst projects
- Optimizing applications with CLI
- Helpful commands

Installing the Command Line Interface

Before continuing with this tutorial, you must first register and download (https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=swg-worklight&S_PKG=ov1268&S_CMP=web_dw_rt_swd) the CLI Installer.

Installation and setup instructions are provided in the download page.
The CLI Installer adds the installation folder to your **path** environment variable so that you can run CLI commands from any directory.

Get the Developer Edition

Install the Command Line Interface

The MobileFirst Platform Command Line Interface (CLI) is provided as an alternative to the Studio IDE. You can create and manage MobileFirst Platform native and hybrid apps, using your own preferred text editors or alternative IDEs.

Before you begin

You must have the Java Developer Kit (JDK) installed on your machine to use the CLI. Additionally, certain operating systems require certain software as a prerequisite. To generate a compatibility report, see Software Product Compatibility Reports.

You must have the JAVA_HOME environment variable set to your JDK directory. For example:

- Windows: C:\Program Files\Java\jdk1.7.0_60

(https://developer.ibm.com/mobilefirstplatform/wp-content/uploads/sites/32/2015/04/cli-instructions.png)

Creating MobileFirst back-end projects

Back-end projects are used for managing hybrid applications, adapters, and other server-side entities. Create a MobileFirst back-end project by using either of these commands:

- `mobilefirst create <project-name>`
- `mfp create <project-name>`

[code lang="xml"]
~/projects $ mfp create MyProject
A MobileFirst project was successfully created at /Users/cli-user/projects/MyProject
[/code]

Managing MobileFirst Server instances

The instance of the Liberty development server is created in the default user directory. For example: `/Users/cli-user/.ibm/mobilefirst/7.1.0.00-build-number/server`.

Server commands

Managing a running server

- `mfp start` - Starts the server.
- `mfp stop` - Stops the server.
- `mfp status` - Gives the status of the server, whether it is running or not, its process ID, and the used port number.
- `mfp config` - Prompts you to select which configuration setting to change. Available options: default browser, preview type (simple or Mobile Browser Simulator), and more.
- `mfp preview [<environments> --noshell|-n]` - Displays a preview of the current application or environment in your default browser.
- `mfp restart` - Restarts the local test server.

- `mfp console` - Opens the MobileFirst Operations Console in your default web browser.

Managing server definitions

- `mfp server add` - Adds a new server definition.
- `mfp server edit` - Edits an existing new server definition.
- `mfp server info` - Returns information about the existing server or about all servers.
- `mfp server remove` - Removes a server definition

Managing client and server artifacts

- `mfp push` - A superset of commands allowing for different actions, depending on the current location
    - If called at the root of a MobileFirst back-end project or a Cordova project, the command registers client and server artifacts (`.wlapps`, `.adapters` and `.war` files) in the server.
    - If called at the root of a native project, the command generates required configuration files and registers the application in the server.
    - Additional `push` commands are also available: `--nosend`, `--password`, `--minify`, and `--concatenate`. Use `mfp help push` to learn more.

Creating hybrid applications and environments

Hybrid applications

Before creating the hybrid application, first navigate to the MobileFirst project directory: `cd MyProject`.
To create a hybrid application, use the `mfp add hybrid <name>` command. The generated hybrid application is placed inside the **apps** folder of the current project.

[code lang="xml"]
~/projects $ cd MyProject
~/projects/MyProject $ mfp add hybrid MyHybridApp
A new hybrid app was added at /Users/cli-user/projects/MyProject/apps/MyHybridApp
[/code]

Environments

Use the `mfp add environment` command to add an environment to the application. You can use this command in the project directory if the project contains only one app. Otherwise, select which app to add the environment to, or call the command from inside the directory of the hybrid app to which the environment should be added.

[code lang="xml"]
~/projects/MyProject $ mfp add environment
[?] To which hybrid app would you want to add environments? (Use the arrow keys.)
❯ MyHybridApp
OtherHybridApp
[/code]

By using the **Interactive Mode**, you can select one or more environments to add:

[code lang="xml"]
~/projects/MyProject/apps/MyHybridApp $ mfp add environment
[?] What environments you want to add to the hybrid app? (Press <space> to select)
❯◯ iPhone
◯ iPad

⬡ Android phone and tablets
⬡ BlackBerry 6 and 7 (deprecated)
⬡ BlackBerry 10
⬡ Windows Phone Silverlight 8
⬡ Windows Universal
(Move up and down to reveal more choices.)
[/code]

To add environments in **Direct Mode**:

[code lang="xml"]
~/projects/MyProject $ mfp add environment iphone,android --app MyHybridApp
A new android environment was added at /Users/cli-user/projects/MyProject/apps/MyHybridApp/android
A new iphone environment was added at /Users/cli-user/projects/MyProject/apps/MyHybridApp/iphone
[/code]

Application skins

To add an application skin from the CLI, use the `mfp add skin` command.

Interactive Mode prompts for the platform to target and name. The current working directory must be under an existing hybrid application, and at least one environment must already be added to this app.

[code lang="xml"]
~/projects/MyProject/apps/MyHybridApp $ mfp add skin
[?] What platform do you want to target? (Use arrow keys)
❯ Android phone and tablets
iPhone
[?] What do you want to be your skin name suffix? Your skin folder name will be 'platform.<suffix>'
A new android skin was added at /Users/cli-user/projects/MyProject/apps/MyHybridApp
[/code]

To add a skin in **Direct Mode**:
[code lang="xml"]mfp add skin [--environment|-e android|blackberry|blackberry10|iphone|ipad <name>]
[/code]

Adding optional features

To add optional features to a hybrid application from the CLI, use the `mfp add feature` command.

Interactive mode prompts for the features to add:
[code lang="xml"]
~/projects/MyProject/apps/MyHybridApp $ mfp add feature
[?] What feature you want to install in this application? NOTE: Features you have already installed are not shown:
FIPS 140-2
IBM Tealeaf SDK
❯ JSONStore

A new jsonstore Feature was added at /Users/cli-user/projects/MyProject/apps/MyHybridApp
[/code]

In **Direct Mode**, you explicitly mention the feature:
[code lang="xml"]
~/projects/MyProject $ cd apps/MyHybridApp/

~/projects/MyProject/apps/MyHybridApp $ mfp add feature jsonstore
A new jsonstore Feature was added at /Users/cli-user/projects/MyProject/apps/MyHybridApp
[/code]

To remove optional features from a hybrid application by using the CLI, use the `mfp remove feature` command.
Interactive mode prompts for the features to remove:

[code lang="xml"]
~/projects/MyProject/apps/MyHybridApp $ mfp remove feature
[?] What feature you want to uninstall from this application? NOTE: Features you have already uninstalled are not shown: (Use arrow keys)
❯ JSONStore
A jsonstore Feature was removed from /Users/cli-user/projects/MyProject/apps/MyHybridApp
[/code]

In **Direct Mode** you explicitly mention the feature:
[code lang="xml"]mfp remove feature [fips|jsonstore|tealeaf][/code]

Concatenation and minification

Applicable only for hybrid applications.

Concatenation is the process of merging web resources such as `.js` and `.css` files reducing the amount of files in the application, while minification is process of *obfuscating* code, making it less readable and more secure.

To concatenate the web resources of a hybrid application, run the command:
[code lang="xml"]
~/projects/MyProject/apps/MyHybridApp $ mfp push --concatenate|-c
[/code]

To concatenate a Hybrid application's web resources, run the command:
[code lang="xml"]
~/projects/MyProject/apps/MyHybridApp $ mfp push --minify|-m
[/code]

Adding MobileFirst Native SDKs

The MobileFirst Native SDK is available for the following environments: iOS, Android, Windows Universal, and Windows Phone Silverlight.
Adding the Native SDK is possible either using locally generated artifacts or by using CocoaPods and Gradle specifically for iOS and Android respectively.

If selecting to add the Native SDK using the locally generated artifacts, first add to the project a NativeAPI "app" using the `mfp add` command.
This "app" is used in the server as a way to identify the client application.

If selecting to add the Native SDK using CocoaPods or Gradle, follow the tutorials in the Configuring the MobileFirst Platform Foundation SDK category (../../configuring-the-mfpf-sdk/) first.
After the SDK is added, navigate to the root of the native project and run the command: `mfp push`.
[code lang="xml"]~/projects/myNativeiOSProject $ mfp push[/code]

Depending on the native project, different artifacts are created:

- `mobilefirst` folder: Contains `.wlapp` files which are pushed and registered in the MobileFirst Server after the command is run.
- `worklight.properties` (for iOS) or `wlclient.properties` (for Android and Windows Universal/Phone)

containing project metadata.

By default, `mfp push` attempts to push and register the artifacts in the default server.
Run `mfp push --nosend|-n` to create the artifacts without also registering in the server.

Creating and testing adapters

To add an adapter, use the `mfp adapter add` command.

Interactive Mode:
[code lang="xml"]
~/projects/MyProject $ mfp adapter add
[?] What do you want to name your MobileFirst Adapter? MyHttpAdapter
[?] What type of adapter would you like? HTTP
[?] Create procedures for offline JSONStore? No
[?] Create procedures for USSD enablement? No
A new HTTP adapter was added at /Users/cli-user/projects/MyProject/adapters/MyHttpAdapter
[/code]

Direct Mode:
`mfp adapter add --type http <adapter-name>`, or `-t` for type. The types of adapters that can be added are HTTP, SQL, Cast Iron, SAP Netweaver Gateway, JMS, SAP Jco, and Java.

[code lang="xml"]
~/projects/MyProject $ mfp adapter add --type http TestAdapter
A new http Adapter was added at /Users/cli-user/projects/MyProject/adapters/TestAdapter
[/code]

Calling an adapter

Use `mfp adapter call` for **Interactive Mode**:

[code lang="xml"]
~/projects/MyProject $ mfp adapter call
[?] Which adapter do you want to use? MyHttpAdapter
[?] Which endpoint do you want to use? MyHttpAdapter/getStories
[?] Enter the comma-separated parameters: "world"
[?] How should the procedure be called? GET

Calling GET '/MyProject/adapters/MyHttpAdapter/getStories?params=["world"]'

Response:
{
"statusCode": 200,
"errors": [],
"isSuccessful": true,
"statusReason": "OK",
"rss": {
"feedburner": "http://rssnamespace.org/feedburner/ext/1.0",
"channel": {
"pubDate": "Thu, 26 Mar 2015 12:36:00 EDT",
"title": "CNN.com - World",
"description": "CNN.com delivers up-to-the-minute news and information on the latest top stories, weather, entertainment, politics and more.",
...
[/code]

Or to call an adapter in **Direct Mode**:
`mfp invoke [<adapter>:<procedure>[\"<json array>\"| --file|-f <path-to-json-array-file>]]`

[code lang="xml"]
~/projects/MyProject $ mfp invoke MyHttpAdapter:getStories \"world\"
{
"statusCode": 200,
"errors": [],
"isSuccessful": true,
"statusReason": "OK",
"rss": {
"feedburner": "http://rssnamespace.org/feedburner/ext/1.0",
"channel": {
"pubDate": "Thu, 26 Mar 2015 12:36:00 EDT",
"title": "CNN.com - World",
"description": "CNN.com delivers up-to-the-minute news and information on the latest top stories, weather, entertainment, politics and more.",
...
[/code]

Exporting and importing MobileFirst projects

Exporting

By using the `export` command, you can create a compressed file, which contains the entire MobileFirst project, or the optimized hybrid assets to use in a native application.

Exporting a project in Interactive Mode
`mfp export` - Running this command in the project root folder prompts you to enter the path and name of the compressed file to export to.
The resulting compressed file is intended to be shared with other MobileFirst developers. The compressed file contains source artifacts and everything another developer would need to build the missing artifacts.

[code lang="xml"]
~/projects/MyProject $ mfp export
[?] Where do you want to export the project? /Users/cli-user/Desktop
[?] What do you want to name your zip project? MyProject
Project successfully exported to /Users/cli-user/Desktop/MyProject.zip
[/code]

Exporting a project in Direct Mode
`mfp export [<path-to-zip-file>]` - If run from the root folder of the project, direct mode takes one additional argument, which is the full path to the compressed file to create.

[code lang="xml"]
~/projects/MyProject $ mfp export /Users/cli-user/Documents/MyProject.zip
Project successfully exported to /Users/cli-user/Documents/MyProject.zip
[/code]

Exporting hybrid assets in Interactive Mode

1. First run the `mfp build` command to make sure that the app is built.
2. Run the hybrid asset export within the environment folder of the existing hybrid app. Interactive mode asks whether to include the native libraries, the path, and file name of the compressed file to export to.

The native libraries are built files. Native application IDEs need to ensure that the hybrid assets can be used in a native app.

[code lang="xml"]
~/projects/MyProject/apps/MyHybridApp/iphone $ mfp export
[?] Would you like to include the native libraries? Yes
[?] Where do you want to export the project? /Users/cli-user/Documents
[?] What do you want to name your zip project? MyiOSHybridApp
Project successfully exported to /Users/cli-user/Documents/MyiOSHybridApp.zip
[/code]

Exporting hybrid assets in Direct Mode
`mfp export [<path-to-zip-file>] [-i | --includeNativeLibs]` - In direct mode, provide the full path to export hybrid assets. If the arguments `-i` or `--includeNativeLibs` are supplied, the native libraries are included.

[code lang="xml"]
~/projects/MyProject/apps/MyHybridApp/iphone $ mfp export /Users/cli-user/Documents/MyiOSHybridApp.zip -i
Project successfully exported to /Users/cli-user/Documents/MyiOSHybridApp.zip
[/code]

Importing CLI-generated projects into MobileFirst Studio

1. From Eclipse, select **File > Import > General > Existing Projects into Workspace** and click **Next**.
2. Select the root folder of the project and click **Finish**.

> For more information, see the topics about the Command Line Interface (CLI) in the user documentation.

Optimizing applications with CLI

You can reduce the size of your application, obfuscate its JavaScript, reduce its load time, or otherwise improve its performance by using minification and concatenation.

> See the "Optimizing applications with CLI" topic in the user documentation

Helpful commands

- `mfp help` – Shows how to use all the commands.
- `mfp help *command-name*` - Shows help for a specific command.
- `mfp info` – Returns the OS release, system memory, `node.js` version, and MobileFirst CLI version.