Installing MobileFirst Server From Command Line tutorial

Overview

Use the command line mode of IBM Installation Manager and Ant tasks to install MobileFirst Server.

Before you begin

- Make sure that one of the following databases and a supported Java version are installed. You also need the corresponding JDBC driver for the database to be available on your computer:
    - Database Management System (DBMS) from the list of supported database:
        - DB2
        - MySQL
        - Oracle

      > **Important:** You must have a database where you can create the tables that are needed by the product, and a database user who can create tables in that database.

      In the tutorial, the steps to create the tables are for DB2. You can find the DB2 installer as a package of IBM MobileFirst Foundation eAssembly on IBM Passport Advantage .
- JDBC driver for your database.
    - For DB2, use the DB2 JDBC driver type 4.
    - For MySQL, use the Connector/J JDBC driver.
    - For Oracle, use the Oracle thin JDBC driver.
- Java 7 or later.
- Download the installer of IBM Installation Manager V1.8.4 or later from Installation Manager and Packaging Utility download links (http://www.ibm.com/support/docview.wss?uid=swg27025142).
- You must also have the installation repository of the MobileFirst Server and the installer of WebSphere Application Server Liberty Core V8.5.5.3 or later. Download these packages from the IBM MobileFirst Foundation eAssembly on Passport Advantage:

MobileFirst Server installation repository
IBM MobileFirst Foundation V8.0 .zip file of Installation Manager Repository for IBM MobileFirst Platform Server

WebSphere Application Server Liberty profile
IBM WebSphere Application Server - Liberty Core V8.5.5.3 or later

Jumpt to

- Creating a database
- Deploying MobileFirst Server to Liberty with Ant tasks
- Testing the installation
- Creating a farm of two Liberty servers that run MobileFirst Server
- Testing the farm and see the changes in MobileFirst Operations Console

Installing IBM Installation Manager

You must install Installation Manager V1.8.4 or later. The older versions of Installation Manager are not able to install IBM MobileFirst Foundation V8.0 because the postinstallation operations of the product require Java 7. The older versions of Installation Manager come with Java 6.

1. Extract the IBM Installation Manager archive file that is downloaded. You can find the installer at Installation Manager and Packaging Utility download links (http://www.ibm.com/support/docview.wss?uid=swg27025142).
2. Review the license agreement for IBM Installation Manager that is in **unzip_IM_1.8.x/license** directory.
3. If you accept the license agreement after the review, install Installation Manager.
    - Run **installc.exe** to install Installation Manager as administrator. Root is needed on Linux or UNIX. On Windows, the administrator privilege is needed. In this mode, the information about the installed packages is placed in a shared location on the disk and any user that is allowed to run Installation Manager can update the applications. The executable file name ends with "c" (**installc**) for a command line installation without a graphical user interface. To install Installation Manager, enter **installc.exe -acceptLicence**.
    - Run **userinstc.exe** to install Installation Manager in user mode. No specific privilege is needed. However, in this mode, the information about the installed packages are placed in the user's home directory. Only that user can update the applications that are installed with Installation Manager. The executable ends with "c" (**userinstc**) for a command line installation without a graphical user interface. To install Installation Manager, enter **userinstc.exe -acceptLicence**.

Installing WebSphere Application Server Liberty Core

The installer for WebSphere Application Server Liberty Core is provided as part of the package for IBM MobileFirst Foundation. In this task, Liberty profile is installed and a server instance is created so that you can install MobileFirst Server on it.

1. Review the license agreement for WebSphere Application Server Liberty Core. The license files can be viewed when you download the installer from Passport Advantage.
2. Extract the compressed file of WebSphere Application Server Liberty Core, that you downloaded, to a folder.

    In the steps that follow, the directory where you extract the installer is referred as **liberty_repository_dir**. It contains a **repository.config** file or a **diskTag.inf** file, among many other files.
3. Decide a directory where Liberty profile is to be installed. It is referred as liberty *install*dir in the next steps.
4. Start a command line and go to **installation_manager_install_dir/tools/eclipse/**.

5. If you accept the license agreement after the review, install Liberty.

   Enter the command: **imcl install com.ibm.websphere.liberty.v85 -repositories liberty_repository_dir -installationDirectory liberty_install_dir -acceptLicense**

   This command installs Liberty in the **liberty_install_dir** directory. The **-acceptLicense** option means that you accept the license terms for the product.
6. Move the directory that contains the servers in a location that does not need specific privileges.

   For the scope of this tutorial, if **liberty_install_dir** points to a location where non-administrator or non-root users cannot modify the files, move the directory that contains the servers to a location that does not need specific privileges. In this way, the installation operations can be done without specific privileges.
   - Go to the installation directory of Liberty.
   - Create a directory named etc. You need administrator or root privileges.
   - In the **etc** directory, create a **server.env** file with the following content: `WLP_USER_DIR=<path to a directory where any user can write>`. For example, on Windows: `WLP_USER_DIR=C:\LibertyServers\usr`.
7. Create a Liberty server that will be used to install the first node of MobileFirst Server at the later part of the tutorial.
   - Start a command line.
   - Go to **liberty_install_dir**/**bin**, and enter **server create mfp1**.

   This command creates a Liberty server instance named **mfp1**. You can see its definition at **liberty_install_dir**/**usr**/**servers**/**mfp1** or **WLP_USER_DIR**/**servers**/**mfp1** (if you modify the directory as described in step 6).

After the server is created, you can start this server with `server start mfp1` from **liberty_install_dir**/**bin**/.
To stop the server, enter the command: `server stop mfp1` from **liberty_install_dir**/**bin**/.

The default home page can be viewed at http://localhost:9080 (http://localhost:9080).

> **Note:** For production, you need to make sure that the Liberty server is started as a service when the host computer starts. Making the Liberty server start as a service is not part of this tutorial.

Installing MobileFirst Server

Make sure that Installation Manager V1.8.4 or later is installed. The installation of MobileFirst Server might not succeed with an older version of Installation Manager because the postinstallation operations require Java 7. The older versions of Installation Manager come with Java 6.

Run Installation Manager to install the binary files of MobileFirst Server on your disk before you create the databases and deploy MobileFirst Server to Liberty profile. During the installation of MobileFirst Server with Installation Manager, an option is proposed to you to install IBM MobileFirst Platform Application Center. Application Center is a different component of the product. For this tutorial, it is not required to be installed with MobileFirst Server.

You also need to specify one property to indicate whether to activate token licensing or not. In this tutorial, it is assumed that token licensing is not needed and the steps to configure MobileFirst Server for token licensing are not included. However, for production installation, you must determine whether you need to

activate token licensing or not. If you do not have a contract to use token licensing with the Rational License Key Server, you do not need to activate token licensing. If you activate token licensing, you must configure MobileFirst Server for token licensing.

In this tutorial, you specify the properties as the parameters through the **imcl** command line. This specification can also be done by using a response file.

1. Review the license agreement for MobileFirst Server. The license files can be viewed when you download the installation repository from Passport Advantage.
2. Extract the compressed file of MobileFirst Server installer, that you downloaded, to a folder.

   In the steps that follow, the directory where you extract the installer is referred as **mfp_repository_dir**. It contains a **MobileFirst_Platform_Server/disk1** folder.
3. Start a command line and go to **installation_manager_install_dir/tools/eclipse/**.
4. If you accept the license agreement after the review in step 1, install MobileFirst Server.

   Enter the command: `imcl install com.ibm.mobilefirst.foundation.server -repositories mfp_repository_dir/MobileFirst_Platform_Server/disk1 -properties user.appserver.selection2=none,user.database.selection2=none,user.database.preinstalled=false,user.licensed.by.tokens=false,user.use.ios.edition=false -acceptLicense`

   The following properties are defined to have an installation without Application Center:

   - **user.appserver.selection2=none**
   - **user.database.selection2=none**
   - **user.database.preinstalled=false**

   This property indicates that token licensing is not activated: **user.licensed.by.tokens=false**.

   Set the value of the **user.use.ios.edition** property to false to install IBM MobileFirst Foundation.

An installation directory that contains the resources to install MobileFirst components is installed.
You can find the resources in the following folders:

- **MobileFirstServer** folder for MobileFirst Server
- **PushService** folder for MobileFirst Server push service
- **ApplicationCenter** folder for Application Center
- **Analytics** folder for MobileFirst Analytics

The goal of this tutorial is to install MobileFirst Server by using the resources in **MobileFirstServer** folder. You can also find some shortcuts for the Server Configuration Tool, Ant, and **mfpadm** program in the **shortcuts** folder.

Creating a database

This task is to ensure that a database exists in your DBMS, and that a user is allowed to use the database, create tables in it, and use the tables. You can skip this task if you plan to use Derby database.

The database is used to store the technical data that is used by the various MobileFirst components:

- MobileFirst Server administration service
- MobileFirst Server live update service
- MobileFirst Server push service
- MobileFirst runtime

In this tutorial, the tables for all the components are placed under the same schema.
Note: The steps in this task are for DB2. If you plan to use MySQL or Oracle, see Database requirements (../../databases/#database-requirements).

1. Log on to the computer that is running the DB2 server. It is assumed that a DB2 user, for example named as **mfpuser**, exists.
2. Verify that this DB2 user has the access to a database with a page size 32768 or more, and is allowed to create implicit schemas and tables in that database.

   By default, this user is a user declared on the operating system of the computer that runs DB2. That is, a user with a login for that computer. If such user exists, the next action in step 3 is not needed.
3. Create a database with the correct page size for this installation if you do not have one.
   - Open a session with a user that has **SYSADM** or **SYSCTRL** permissions. For example, use the user **db2inst1** that is the default admin user that is created by the DB2 installer.
   - Open a DB2 command line processor:
     - On Windows systems, click **Start → IBM DB2 → Command Line Processor**.
     - On Linux or UNIX systems, go to ~/**sqllib**/**bin** (or **db2_install_dir**/**bin** if sqllib is not created in the administrator's home directory) and enter `./db2`.
   - Enter the following SQL statements to create a database that is called **MFPDATA**:

     ```
     CREATE DATABASE MFPDATA COLLATE USING SYSTEM PAGESIZE 32768
     CONNECT TO MFPDATA
     GRANT CONNECT ON DATABASE TO USER mfpuser
     GRANT CREATETAB ON DATABASE TO USER mfpuser
     GRANT IMPLICIT_SCHEMA ON DATABASE TO USER mfpuser
     DISCONNECT MFPDATA
     QUIT
     ```

   If you defined a different user name, replace **mfpuser** with your own user name.

   > **Note:** The statement does not remove the default privileges granted to PUBLIC in a default DB2 database. For production, you might need to reduce the privileges in that database to the minimum requirement for the product. For more information about DB2 security and an example of the security practices, see DB2 security, Part 8: Twelve DB2 security best practices (http://www.ibm.com/developerworks/data/library/techarticle/dm-0607wasserman/).

Deploying MobileFirst Server to Liberty with Ant tasks

You use the Ant tasks to run the following operations:

- Create the tables in the database that are needed by the MobileFirst applications
- Deploy the web applications of MobileFirst Server (the runtime, administration service, live update service, push service components, and MobileFirst Operations Console) to Liberty server.

The following MobileFirst applications are not deployed by Ant tasks:

MobileFirst Analytics

MobileFirst Analytics is typically deployed on a different set of servers than MobileFirst Server because of its high memory requirements. MobileFirst Analytics can be installed manually or with Ant tasks. If it is already installed, you can enter its URL, the user name, and password to send data to it in the Server

Configuration Tool. The Server Configuration Tool then configures the MobileFirst apps to send data to MobileFirst Analytics.

Application Center

This application can be used to distribute mobile apps internally to the employees that use the apps, or for test purpose. It is independent of MobileFirst Server and is not necessary to install together with MobileFirst Server.

Pick the appropriate XML file that contains the Ant tasks and configure the properties.

- Make a copy of the **mfp_install_dir/MobileFirstServer/configuration-samples/configure-liberty-db2.xml** file to a working directory. This file contains the Ant tasks for installing MobileFirst Server on Liberty with DB2 as the database. Before you use it, define the properties to describe where the applications of MobileFirst Server are to be deployed.
- Edit the copy of the XML file and set the values of the following properties:
    - **mfp.admin.contextroot** to /**mfpadmin**
    - **mfp.runtime.contextroot** to /**mfp**
    - **database.db2.host** to the value to the host name of the computer that runs your DB2 database. If the database is on the same computer as Liberty, use **localhost**.
    - **database.db2.port** to the port to which the DB2 instance is listening. By default, it is **50000**.
    - **database.db2.driver.dir** to the directory that contains your DB2 driver: **db2jcc4.jar** and **db2jcc_license_cu.jar**. In a standard DB2 distribution, these files are found in **db2_install_dir/java**.
    - **database.db2.mfp.dbname** to **MFPDATA** - the database name that you create in Creating a database.
    - **database.db2.mfp.schema** to **MFPDATA** - the value of the schema where the tables for MobileFirst Server are to be created. If your DB user is not able to create a schema, set the value to an empty string. For example, **database.db2.mfp.schema=""**.
    - **database.db2.mfp.username** to the DB2 user that creates the tables. This user also uses the tables at run time. For this tutorial, use **mfpuser**.
    - **appserver.was.installdir** to the Liberty installation directory.
    - **appserver.was85liberty.serverInstance** to **mfp1** - the value to the name of the Liberty server where MobileFirst Server is to be installed.
    - **mfp.farm.configure** to **false** to install MobileFirst Server in stand-alone mode.
    - **mfp.analytics.configure** to **false**. The connection to MobileFirst Analytics is not in the scope of this tutorial. You can ignore the other properties mfp.analytics.****.
    - **mfp.admin.client.id** to **admin-client-id**.
    - **mfp.admin.client.secret** to **adminSecret** (or choose another secret password).
    - **mfp.push.client.id** to **push-client-id**.
    - **mfp.push.client.secret** to **pushSecret** (or choose another secret password).
    - **mfp.config.admin.user** to the user name of the MobileFirst Server live update service. In a server farm topology, the user name must be the same for all the members of the farm.
    - **mfp.config.admin.password** to the password of the MobileFirst Server live update service. In a server farm topology, the password must be the same for all the members of the farm.
- Keep the default values of the following properties as-is:
    - **mfp.admin.console.install** to true

    - **mfp.admin.default.user** to **admin** - the name of a default user that is created to log in to MobileFirst Operations Console.
    - **mfp.admin.default.user.initialpassword** to **admin** - the password of a default user that is created to log in to the admin console.
    - **appserver.was.profile** to **Liberty**. If the value is different, the Ant task assumes that the installation is on a WebSphere Application Server server.
- Save the file after the properties are defined.
- Run `mfp_server_install_dir/shortcuts/ant -f configure-liberty-db2.xml` to show a list of possible targets for the Ant file.
- Run `mfp_server_install_dir/shortcuts/ant -f configure-liberty-db2.xml databases` to create the database tables.
- Run `mfp_server_install_dir/shortcuts/ant -f configure-liberty-db2.xml install` to install MobileFirst Server.

> **Note:** If you do not have DB2, and want to test the installation with an embedded Derby as a database, use the **mfp_install_dir/MobileFirstServer/configuration-samples/configure-liberty-derby.xml** file. However, you cannot do the last step of this tutorial (Creating a farm of two Liberty servers that run MobileFirst Server) because the Derby database cannot be accessed by multiple Liberty servers. You must set the properties except the DB2 related ones (**database.db2**, ...). For Derby, set the value of the property **database.derby.datadir** to the directory where Derby database can be created. Also, set the value of the property **database.derby.mfp.dbname** to **MFPDATA**.

The following operations are run by the Ant tasks:

1. The tables for the following components are created in the database:
    - The administration service and the live update service. Created by the `admdatabases` Ant target.
    - The runtime component. Created by the `rtmdatabases` Ant target.
    - The push service. Created by the `pushdatabases` Ant target.
2. The WAR files of the various components are deployed to Liberty server. You can see the details of the operations in the log under `adminstall`, `rtminstall`, and `pushinstall` targets.

If you have access to the DB2 server, you can list the tables that are created by using these instructions:

1. Open a DB2 command line processor with mfpuser as described in step 3 of Creating a database.
2. Enter the SQL statements:

```
CONNECT TO MFPDATA USER mfpuser USING mfpuser_password
LIST TABLES FOR SCHEMA MFPDATA
DISCONNECT MFPDATA
QUIT
```

Take note of the following database factors:

Database user consideration

In the Server Configuration Tool, only one database user is needed. This user is used to create the tables, but is also used as the data source user in the application server at run time. In production environment, you might want to restrict the privileges of the user that is used at run time to the strict minimum (`SELECT`

`/ INSERT / DELETE / UPDATE)`, and thus provide a different user for deployment in the application server. The Ant files that are provided as examples also use the same users for both cases. However, in the case of DB2, you might want to create your own versions of files. As such, you can distinguish the user that is used to create the databases from the user that is used for the data source in the application server with the Ant tasks.

Database tables creation

For production, you might want to create the tables manually. For example, if your DBA wants to override some default settings or assign specific table spaces. The database scripts that are used to create the tables are available in **mfp_server_install_dir/MobileFirstServer/databases** and **mfp_server_install_dir/PushService/databases**. For more information, see Creating the database tables manually (../../databases/#create-the-database-tables-manually).

The **server.xml** file and some application server setting are modified during the installation. Before each modification, a copy of the **server.xml** file is made, such as **server.xml.bak**, **server.xml.bak1**, and **server.xml.bak2**. To see everything that was added, you can compare the **server.xml** file with the oldest backup (server.xml.bak). On Linux, you can use the command diff `--strip-trailing-cr server.xml server.xml.bak` to see the differences. On AIX , use the command `diff server.xml server.xml.bak` to find the differences.

Modification of the application server settings (specific to Liberty):

1. The Liberty features are added.

   The features are added for each application and can be duplicated. For example, the JDBC feature is used for both the administration service and the runtime components. This duplication allows the removal of the features of an application when it is uninstalled without breaking the other applications. For example, if you decide at some point to uninstall the push service from a server and install it on another server. However, not all topologies are possible. The administration service, the live update service, and the runtime component must be on the same application server with Liberty profile. For more information, see Constraints on MobileFirst Server administration service, MobileFirst Server live update service and MobileFirst runtime (../../topologies/#constraints-on-mobilefirst-server-administration-service-mobilefirst-server-live-update-service-and-mobilefirst-runtime). The duplication of features does not create issue unless the features that added are conflicting. Adding the jdbc-40 and jdbc-41 features would cause a problem, but adding twice the same feature does not.
2. `host='*'` is added in the `httpEndPoint` declaration.

   This setting is to allow the connection to the server from all network interfaces. In production, you might want to restrict the host value of the HTTP endpoint.
3. The **tcpOptions** element (**tcpOptions soReuseAddr="true"**) is added in the server configuration to enable immediate rebind to a port with no active listener and improve the throughput of the server.
4. A keystore with ID **defaultKeyStore** is created if it does not exist.

   he keystore is to enable the HTTPS port and more specifically, to enable the JMX communication between the administration service (mfp-admin-service.war) and the runtime component (mfp-server.war). The two applications communicate via JMX. In the case of Liberty profile, restConnector is used to communicate between the applications in a single server and also between the servers of a Liberty Farm. It requires the use of HTTPS. For the keystore that is created by default, Liberty profiles creates a certificate with a validity period of 365 days. This configuration is not intended for production use. For production, you need to reconsider to use your

own certificate.

To enable JMX, a user with administrator role (named as MfpRESTUser) is created in the basic registry. Its name and password are provided as JNDI properties (mfp.admin.jmx.user and mfp.admin.jmx.pwd) and are used by the runtime component and the administration service to run JMX queries. In the global JMX properties, some properties are used to define the cluster mode (stand-alone server or working in a farm). The Server Configuration Tool sets the mfp.topology.clustermode property to Standalone in Liberty server. In the later part of this tutorial about the creation of a farm, the property is modified to Cluster.

5. The creation of users (Also valid for Apache Tomcat and WebSphere Application Server)
    - Optional Users: The Server Configuration Tool creates a test user (admin/admin) so that you can use this user to log to the console after the installation.
    - Mandatory Users: The Server Configuration Tool also creates a user (named as configUser_mfpadmin with a randomly generated password) to be used by the administration service to contact the local live update service. For Liberty server, MfpRESTUser is created. If your application server is not configured to use a basic registry (for example, an LDAP registry), the Server Configuration Tool is unable to request the name of an existing user. In this case, you need to use Ant tasks.
6. The **webContainer** element is modified.

   The `deferServletLoad` web container custom property is set to false. Both the runtime component and the administration service must start when the server starts. These components can thus register the JMX beans and start the synchronization procedure that allows the runtime component to download all the applications and adapters that it needs to serve.
7. The default executor is customized to set large values to `coreThreads` and `maxThreads` if you use Liberty V8.5.5.5 or earlier. The default executor is automatically tuned by Liberty as of V8.5.5.6.

   This setting avoids timeout issues that break the startup sequence of the runtime component and administration service on some Liberty versions. The absence of this statement can be the cause of these errors in the server log file:

   > Failed to obtain JMX connection to access an MBean. There might be a JMX configuration error: Read timed out FWLSE3000E: A server error was detected. FWLSE3012E: JMX configuration error. Unable to obtain MBeans. Reason: "Read timed out".

Declaration of applications

The following applications are installed:

- **mfpadmin**, the administration service
- **mfpadminconfig**, the live update service
- **mfpconsole**, MobileFirst Operations Console
- **mobilefirs**t, MobileFirst runtime component
- **imfpush**, the push service

The Server Configuration Tool installs all the applications on the same server. You can separate the applications in different application servers, but under certain constraints that are documented in Topologies and network flows (../../topologies).

For an installation on different servers, you cannot use the Server Configuration Tool. Use Ant tasks or install the product manually.

Administration service

The administration service is the service for managing MobileFirst applications, adapters, and their configurations. It is secured by security roles. By default, the Server Configuration Tool adds a user (admin) with the administrator role, that you can use to log in to the console for testing. The configuration of the security role must be done after an installation with the Server Configuration Tool (or with Ant tasks). You might want to map the users or the groups that come from the basic registry or an LDAP registry that you configure in your application server to each security role.

The class loader is set with delegation parent last for Liberty profile and WebSphere Application Server, and for all MobileFirst applications. This setting is to avoid conflicts between the classes packaged in the MobileFirst applications and the classes of the application server. Forgetting to set the class loader delegation to parent last is a frequent source of error in manual installation. For Apache Tomcat, this declaration is not needed.

In Liberty profile, a common library is added to the application for decrypting passwords that are passed as JNDI properties. The Server Configuration Tool defines two mandatory JNDI properties for the administration service: **mfp.config.service.user** and **mfp.config.service.password**. They are used by the administration service to connect to the live update service with its REST API. More JNDI properties can be defined to tune the application or adapt it to your installation particularities. For more information, see List of JNDI properties for MobileFirst Server administration service (../../server-configuration/#list-of-jndi-properties-for-mobilefirst-server-administration-service).

The Server Configuration Tool also defines the JNDI properties (the URL and the OAuth parameters to register the confidential clients) for the communication with the push service.
The data source to the database that contains the tables for the administration service is declared, as well as a library for its JDBC driver.

Live update service

The live update service stores information about the runtime and application configurations. It is controlled by the administration service and must always run on the same server as the administration service. The context root is **context_root_of_admin_serverconfig**. As such, it is **mfpadminconfig**. The administration service assumes that this convention is respected to create the URL of its requests to the REST services of the live update service.

The class loader is set with delegation parent last as discussed in the administration service section.

The live update service has one security role, **admin_config**. A user must be mapped to that role. Its password and login must be provided to the administration service with the JNDI property: **mfp.config.service.user** and **mfp.config.service.password**. For information about the JNDI properties, see List of JNDI properties for MobileFirst Server administration service (../../server-configuration/#list-of-jndi-properties-for-mobilefirst-server-administration-service) and List of JNDI properties for MobileFirst Server live update service (../../server-configuration/#list-of-jndi-properties-for-mobilefirst-server-live-update-service).

It also needs a data source with JNDI name on Liberty profile. The convention is **context_root_of_config_server/jdbc/ConfigDS**. In this tutorial, it is defined as **mfpadminconfig/jdbc/ConfigDS**. In an installation by the Server Configuration Tool or with Ant tasks, the tables of the live update service are in the same database and schema as the tables of the administration service. The user to access these tables is also the same.

MobileFirst Operations Console

MobileFirst Operations Console is declared with the same security roles as the administration service. The users that are mapped to the security roles of MobileFirst Operations Console must also be mapped to the

same security role of the administration service. Indeed, MobileFirst Operations Console runs queries to the administration service on the behalf of the console user.

The Server Configuration Tool positions one JNDI property, **mfp.admin.endpoint**, that indicates how the console connects to the administration service. The default value set by the Server Configuration Tool is `*://*:*/mfpadmin`. The setting means that it must use the same protocol, host name, and port as the incoming HTTP request to the console, and the context root of the administration service is /mfpadmin. If you want to force the request to go though a web proxy, change the default value. For more information about the possible values for this URL, or for information about other possible JNDI properties, see List of JNDI properties for MobileFirst Server administration service (../../server-configuration/#list-of-jndi-properties-for-mobilefirst-server-administration-service).

The class loader is set with delegation parent last as discussed in the administration service section.

MobileFirst runtime

This application is not secured by a security role. It is not required to log in with a user known by the Liberty server, to access this application. The mobile devices requests are routed to the runtime. They are authenticated by other mechanisms specific to the product (such as OAuth) and the configuration of the MobileFirst applications.

The class loader is set with delegation parent last as discussed in the administration service section.

It also needs a data source with JNDI name on Liberty profile. The convention is **context_root_of_runtime/jdbc/mfpDS**. In this tutorial, it is defined as **mobilefirst/jdbc/mfpDS**. In an installation by the Server Configuration Tool or with Ant tasks, the tables of the runtime are in the same database and schema as the tables of the administration service. The user to access these tables is also the same.

Push service

This application is secured by OAuth. The valid OAuth tokens must be included in any HTTP request to the service.

The configuration of OAuth is made through the JNDI properties (such as the URL of the authorization server, the client ID, and the password of the push service). The JNDI properties also indicate the security plug-in (**mfp.push.services.ext.security**) and the fact that a relational database is used (**mfp.push.db.type**). The requests from the mobile devices to the push service are routed to this service. The context root of the push service must be /imfpush. The client SDK computes the URL of the push service based on the URL of the runtime with the context root (/**imfpush**). If you want to install the push service on a different server than the runtime, you need to have an HTTP router that can route the device requests to the relevant application server.

The class loader is set with delegation parent last as discussed in the administration service section.

It also needs a data source with JNDI name on Liberty profile. The JNDI name is **imfpush/jdbc/imfPushDS**. In an installation by the Server Configuration Tool or with Ant tasks, the tables of the push service are in the same database and schema as the tables of the administration service. The user to access these tables is also the same.

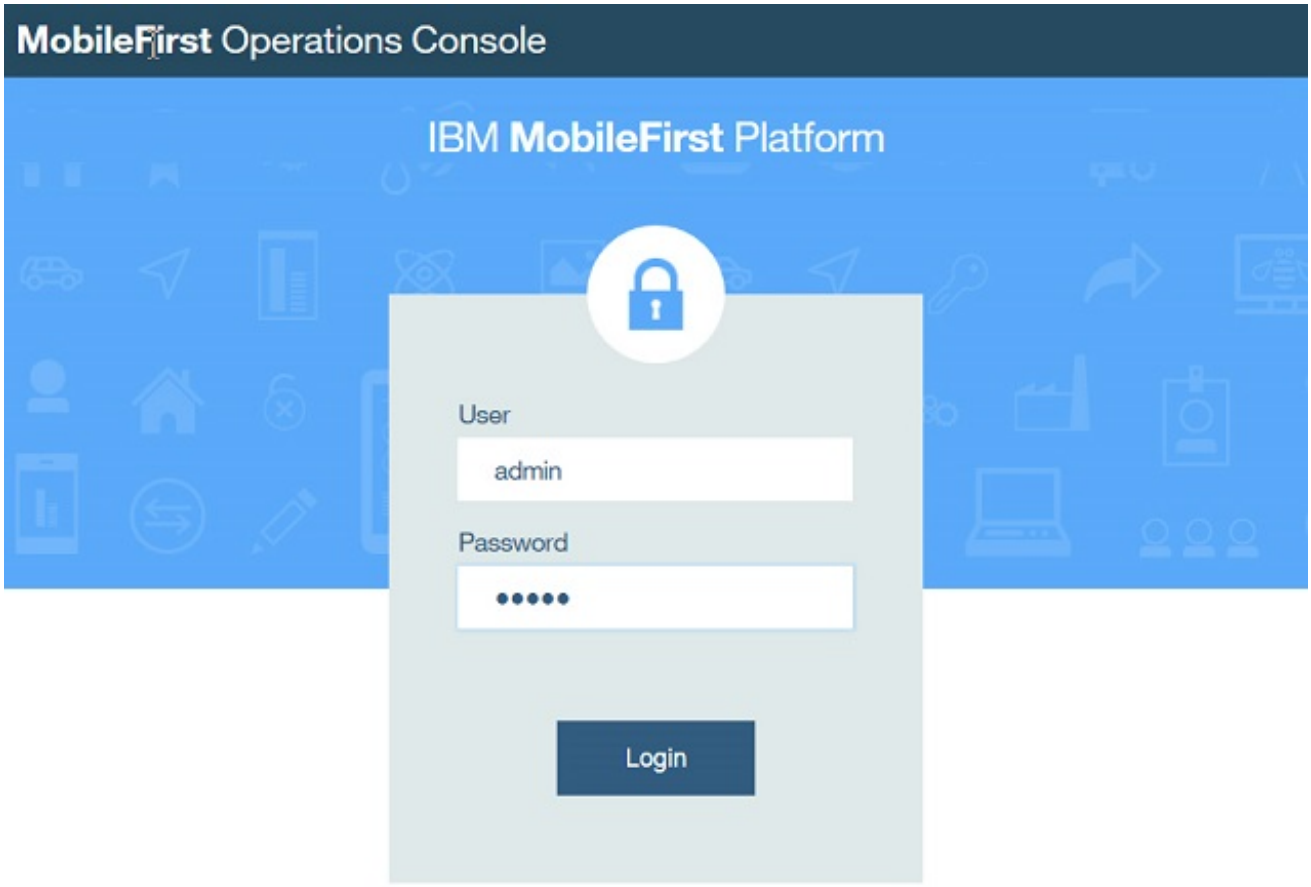
Other files modification

The Liberty profile **jvm.options** file is modified. A property (**com.ibm.ws.jmx.connector.client.rest.readTimeout**) is defined to avoid timeout issues with JMX when the runtime synchronizes with the administration service.

Testing the installation

After the installation is complete, you can use this procedure to test the components that are installed.

1. Start the server by using the command **server start mfp1**. The binary file for the server is in **liberty_install_dir/bin**.
2. Test MobileFirst Operations Console with a web browser. Go to http://localhost:9080/mfpconsole (http://localhost:9080/mfpconsole). By default, the server runs on port 9080. However, you can verify the port in the element `<httpEndpoint>` as defined in the **server.xml** file. A login screen is displayed.



1. Log in with **admin/admin**. This user is created by default by the Server Configuration Tool.

   > **Note:** If you connect with HTTP, the login ID and password are sent in clear text in the network. For a secure login, use HTTPS to log to the server. You can see the HTTPS port of the Liberty server in the httpsPort attribute of the `<httpEndpoint>` element in the **server.xml** file. By default, the value is 9443.

2. Log out of the console with **Hello Admin → Sign Out**.
3. Enter the following URL: https://localhost:9443/mfpconsole (https://localhost:9443/mfpconsole) in the web browser and accept the certificate. By default, the Liberty server generates a default certificate that is not known by your web browser, you need to accept the certificate. Mozilla Firefox presents this certification as a security exception.
4. Log in again with **admin/admin**. The login and password are encrypted between your web browser and MobileFirst Server. In production, you might want to close the HTTP port.

Creating a farm of two Liberty servers that run MobileFirst Server

In this task, you will create a second Liberty server that runs the same MobileFirst Server and connected to

the same database. In production, you might use more than one server for performance reasons, to have enough servers to serve the number of transactions per second that is needed for your mobile applications at peak time. It is also for high availability reasons to avoid having a single point of failure.

When you have more than one server that runs MobileFirst Server, the servers must be configured as a farm. This configuration enables any administration service to contact all the runtimes of a farm. If the cluster is not configured as a farm, only the runtime that runs in the same application server as the management service that runs the management operation is notified. Others runtimes are not aware of the change. For example, you deploy a new version of an adapter in a cluster that is not configured as a farm, only one server would serve the new adapter. The other servers would continue to serve the old adapter. The only situation where you can have a cluster and do not need to configure a farm is when you install your servers on WebSphere Application Server Network Deployment. The administration service is able to find all the servers by querying the JMX beans with the deployment manager. The deployment manager must be running to allow management operations because it is used to provide the list of the MobileFirst JMX beans of the cell.

When you create a farm, you also need to configure an HTTP server to send queries to all the members of the farm. The configuration of an HTTP server is not included in this tutorial. This tutorial is only about configuring the farm so that management operations are replicated to all the runtime components of the cluster.

1. Create a second Liberty server on the same computer.
    - Start a command line.
    - Go to **liberty_install_dir**/**bin**, and enter server create **mfp2**.
2. Modify the HTTP and HTTPS ports of the server **mfp2** so that they do not conflict with the ports of server **mfp1**.
    - Go to the second server directory.

      The directory is **liberty_install_dir**/**usr**/**servers**/**mfp2** or **WLP_USER_DIR**/**servers**/**mfp2** (if you modify the directory as described in step 6 of Installing WebSphere Application Server Liberty Core).
    - Edit the **server.xml** file. Replace

      ```
      <httpEndpoint id="defaultHttpEndpoint"
      httpPort="9080"
      httpsPort="9443" />
      ```

      With:

      ```
      <httpEndpoint id="defaultHttpEndpoint"
      httpPort="9081"
      httpsPort="9444" />
      ```

      The HTTP and HTTPS ports of the server mfp2 do not conflict with the ports of the server mfp1 with this change. Make sure to modify the ports before you run the installation of MobileFirst Server. Otherwise, if you modify the port after the installation is made, you also need to reflect the change of the port in the JNDI property: **mfp.admin.jmx.port**.
3. Copy the Ant file that you used in Deploying MobileFirst Server to Liberty with Ant tasks, and change the value of the property **appserver.was85liberty.serverInstance** to **mfp2**. The Ant tasks detect that the databases exist and do not create the tables (see the following log extract). Then, the applications are deployed to the server.

```
[configuredatabase] Checking connectivity to MobileFirstAdmin database MFPDATA with schema '
MFPDATA' and user 'mfpuser'...
[configuredatabase] Database MFPDATA exists.
[configuredatabase] Connection to MobileFirstAdmin database MFPDATA with schema 'MFPDATA'
and user 'mfpuser' succeeded.
[configuredatabase] Getting the version of MobileFirstAdmin database MFPDATA...
[configuredatabase] Table MFPADMIN_VERSION exists, checking its value...
[configuredatabase] GetSQLQueryResult => MFPADMIN_VERSION = 8.0.0
[configuredatabase] Configuring MobileFirstAdmin database MFPDATA...
[configuredatabase] The database is in latest version (8.0.0), no upgrade required.
[configuredatabase] Configuration of MobileFirstAdmin database MFPDATA succeeded.
```

1. Test the two servers with HTTP connection.
   - Open a web browser.
   - Enter the following URL: http://localhost:9080/mfpconsole (http://localhost:9080/mfpconsole). The console is served by server mfp1.
   - Log in with **admin**/**admin**.
   - Open a tab in the same web browser and enter the URL: http://localhost:9081/mfpconsole (http://localhost:9081/mfpconsole). The console is served by server mfp2.
   - Log in with admin/admin. If the installation is done correctly, you can see the same welcome page in both tabs after login.
   - Return to first browser tab and click **Hello, admin → Download Audit Log**. You are logged out of the console and see the login screen again. This logout behavior is an issue. The problem happens because when you log on to server mfp2, a Lightweight Third Party Authentication (LTPA) token is created and stored in your browser as a cookie. However, this LTPA token is not recognized by server mfp1. Switching between servers is likely to happen in a production environment when you have an HTTP load balancer in front of the cluster. To resolve this issue, you must ensure that both servers (mfp1 and mfp2) generate the LTPA tokens with the same secret keys. Copy the LTPA keys from server mfp1 to server mfp2.
   - Stop both servers with these commands:

     ```
     server stop mfp1
     server stop mfp2
     ```

   - Copy the LTPA keys of server mfp1 to server mfp2. From **liberty_install_dir**/**usr**/**servers** or **WLP_USER_DIR**/**servers**, run the following command depending on your operating system.
     - On UNIX: `cp mfp1/resources/security/ltpa.keys mfp2/resources/security/ltpa.keys`
     - On Windows: `copy mfp1/resources/security/ltpa.keys mfp2/resources/security/ltpa.keys`
   - Restart the servers. Switch from one browser tab to another other does not require you to relogin. In a Liberty server farm, all servers must have the same LTPA keys.
2. Enable the JMX communication between the Liberty servers.

   The JMX communication with Liberty, is done via the Liberty REST connector over the

HTTPS protocol. To enable this communication, each server of the farm must be able to recognize the SSL certificate of the other members. You need to exchange the HTTPS certificates in their truststores. Use IBM utilities such as Keytool, which is part of the IBM JRE distribution in **java/bin** to configure the truststore. The locations of the keystore and truststore are defined in the **server.xml** file. By default, the keystore of Liberty profile is at **WLP_USER_DIR/servers/server_name/resources/security/key.jks**. The password of this default keystore, as can be seen in the **server.xml** file, is **mobilefirst**.

> **Tip:** You can change it with the Keytool utility, but you must also change the password in the server.xml file so that Liberty server can read that keystore. In this tutorial, use the default password.

- In **WLP_USER_DIR/servers/mfp1/resources/security**, enter `keytool -list -keystore key.jks`. The command shows the certificates in the keystore. There is only one named **default**. You are prompted for the password of the keystore (mobilefirst) before you can see the keys. This is the case for all the next commands with Keytool utility.
- Export the default certificate of server mfp1 with the command: `keytool -exportcert -keystore key.jks -alias default -file mfp1.cert`.
- In **WLP_USER_DIR/servers/mfp2/resources/security**, export the default certificate of server mfp2 with the command: `keytool -exportcert -keystore key.jks -alias default -file mfp2.cert`.
- In the same directory, import the certificate of server mfp1 with the command: `keytool -import -file ../../../mfp1/resources/security/mfp1.cert -keystore key.jks`. The certificate of server mfp1 is imported into the keystore of server mfp2 so that server mfp2 can trust the HTTPS connections to server mfp1. You are asked to confirm that you trust the certificate.
- In **WLP_USER_DIR/servers/mfp1/resources/security**, import the certificate of server mfp2 with the command: `keytool -import -file ../../../mfp2/resources/security/mfp2.cert -keystore key.jks`. After this step, the HTTPS connections between the two servers are possible.

Testing the farm and see the changes in MobileFirst Operations Console

1. Start the two servers:

```
server start mfp1
server start mfp2
```

2. Access the console. For example, http://localhost:9080/mfpconsole (http://localhost:9080/mfpconsole), or https://localhost:9443/mfpconsole (https://localhost:9443/mfpconsole) in HTTPS. In the left sidebar, an extra menu that is labeled as **Server Farm Nodes** appears. If you click **Server Farm Nodes**, you can the status of each node. You might need to wait a bit for both nodes to be started.

Last modified on