Installing MobileFirst Server to an application server

Overview

The installation of the components can be done by using Ant Tasks, the Server Configuration Tool, or manually. Find out the prerequisite and the details about the installation process so that you can install the components on the application server successfully.

Before you proceed with installing the components to the application server, ensure that the databases and the tables for the components are prepared and ready to use. For more information, see Setting up databases (../databases).

The server topology to install the components must also be defined. See Topologies and network flows (../topologies).

Jump to

Application server prerequisites

Depending on your choice of the application server, select one of the following topics to find out the prerequisites that you must fulfill before you install the MobileFirst Server components.

- Apache Tomcat prerequisites
- WebSphere Application Server Liberty prerequisites
- WebSphere Application Server and WebSphere Application Server Network Deployment prerequisites

Apache Tomcat prerequisites

MobileFirst Server has some requirements for the configuration of Apache Tomcat that are detailed in the following topics.
Ensure that you fulfill the following criteria:

- Use a supported version of Apache Tomcat. See System requirements (../../../product-overview/requirements).
- Apache Tomcat must be run with JRE 7.0 or later.
- The JMX configuration must be enabled to allow the communication between the administration service and the runtime component. The communication uses RMI as described in **Configuring JMX connection for Apache Tomcat** below.

Click for instructions on configuring JMX connection for Apache Tomcat

WebSphere Application Server Liberty prerequisites

IBM MobileFirst Server has some requirements for the configuration of the Liberty server that are detailed in the following topics.
Ensure that you fulfill the following criteria:

- Use a supported version of Liberty. See System requirements (../../../product-overview/requirements).
- Liberty must be run with JRE 7.0 or later. JRE 6.0 is not supported.
- Some versions of Liberty support both the features of Java EE 6 and Java EE 7. For example, jdbc-4.0 Liberty feature is part of Java EE 6, whereas jdbc-4.1 Liberty feature is part of Java EE 7. MobileFirst Server V8.0.0 can be installed with Java EE 6 or Java EE 7 features. However, if you want to run an older version of MobileFirst Server on the same Liberty server, you must use the Java EE 6 features. MobileFirst Server V7.1.0 and earlier, does not support the Java EE 7 features.
- JMX must be configured as documented in **Configuring JMX connection for WebSphere Application Server Liberty profile** below.
- For an installation in a production environment, you might want to start the Liberty server as a service on Windows, Linux, or UNIX systems so that: The MobileFirst Server components are started automatically when the computer starts. The process that runs Liberty server is not stopped when the user, who started the process, logs out.
- MobileFirst Server V8.0.0 cannot be deployed in a Liberty server that contains the deployed MobileFirst Server components from the previous versions.
- For an installation in a Liberty collective environment, the Liberty collective controller and the Liberty collective cluster members must be configured as documented in Configuring a Liberty collective (http://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.wlp.nd.doc/ae/tagt_wlp_configure_collective.html?

view=kc).

Click for instructions on configuring JMX connection for WebSphere Application Server Liberty profile

WebSphere Application Server and WebSphere Application Server Network Deployment prerequisites

IBM MobileFirst Server has some requirements for the configuration of WebSphere Application Server and WebSphere Application Server Network Deployment that are detailed in the following topics.
Ensure that you fulfill the following criteria:

- Use a supported version of WebSphere Application Server. See System requirements (../../../product-overview/requirements).
- The application server must be run with JRE 7.0. By default, WebSphere Application Server uses Java 6.0 SDK. To switch to Java 7.0 SDK, see Switching to Java 7.0 SDK in WebSphere Application Server (https://www.ibm.com/support/knowledgecenter/SSWLGF_8.5.5/com.ibm.sr.doc/twsr_java17.html).
- The administrative security must be turned on. MobileFirst Operations Console, the MobileFirst Server administration service, and the MobileFirst Server configuration service are protected by security roles. For more information, see Enabling security (https://www.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/tsec_csec2.html?cp=SSEQTP_8.5.5%2F1-8-2-31-0-2&lang=en).
- The JMX configuration must be enabled to allow the communication between the administration service and the runtime component. The communication uses SOAP. For WebSphere Application Server Network Deployment, RMI can be used. For more information, see **Configuring JMX connection for WebSphere Application Server and WebSphere Application Server Network Deployment** below.

Click for instructions on configuring JMX connection for WebSphere Application Server and WebSphere Application Server Network Deployment

File system prerequisites

To install IBM MobileFirst Server to an application server, the MobileFirst installation tools must be run by a user that has specific file system privileges.
The installation tools include:

- IBM Installation Manager
- The Server Configuration Tool
- The Ant tasks to deploy MobileFirst Server

For WebSphere Application Server Liberty profile, you must have the required permission to perform the following actions:

- Read the files in the Liberty installation directory.
- Create files in the configuration directory of the Liberty server, which is typically usr/servers/server-name, to create backup copies and modify server.xml and jvm.options.
- Create files and directories in the Liberty shared resource directory, which is typically usr/shared.
- Create files in the Liberty server apps directory, which is typically usr/servers/server-name/apps.

For WebSphere Application Server full profile and WebSphere Application Server Network Deployment, you must have the required permission to perform the following actions:

- Read the files in the WebSphere Application Server installation directory.
- Read the configuration file of the selected WebSphere Application Server full profile or of the Deployment Manager profile.
- Run the wsadmin command.
- Create files in the profiles configuration directory. The installation tools put resources such as shared libraries or JDBC drivers in that directory.

For Apache Tomcat, you must have the required permission to perform the following actions:

- Read the configuration directory.
- Create backup files and modify files in the configuration directory, such as server.xml, and tomcat-users.xml.
- Create backup files and modify files in the bin directory, such as setenv.bat.
- Create files in the lib directory.
- Create files in the webapps directory.

For all these application servers, the user who runs the application server must be able to read the files that were created by the user who ran the MobileFirst installation tools.

Installing with the Server Configuration Tool

Use the Server Configuration Tool to install the MobileFirst Server components to your application server.

The Server Configuration Tool can set up the database and install the components to an application server. This tool is meant for a single user. The configuration files are store on the disk. The directory where they are stored can be modified with menu **File → Preferences**. The files must be used only by one instance of the Server Configuration Tool at the time. The tool does not manage concurrent access to the same file. If you have multiple instances of the tool accessing the same file, the data might be lost. For more information about how the tool creates and setup the databases, see Create the database tables with the Server Configuration Tool (../databases/#create-the-database-tables-with-the-server-configuration-tool). If the databases exist, the tool can detect them by testing the presence and the content of some test tables and does not modify these database tables.

- Supported operating systems
- Supported topologies
- Running the Server Configuration Tool
- Applying a fix pack by using the Server Configuration Tool

Supported operating systems

You can use the Server Configuration Tool if you are on the following operating systems:

- Windows x86 or x86-64
- macOS x86-64
- Linux x86 or Linux x86-64

The tool is not available on other operating systems. You need to use Ant tasks to install the MobileFirst Server components as described in Installing with Ant Tasks.

Supported topologies

The Server Configuration Tool installs the MobileFirst Server components with the following topologies:

- All components (MobileFirst Operations Console, the MobileFirst Server administration service, the MobileFirst Server live update service, and the MobileFirst runtime) are in the same application server. However, on WebSphere Application Server Network Deployment when you install on a cluster, you can specify a different cluster for the administration and live update services, and for the runtime. On Liberty collective, MobileFirst Operations Console, the administration service, and the live update service are installed in a collective controller and the runtime in a collective member.
- If the MobileFirst Server push service is installed, it is also installed on the same server. However, on WebSphere Application Server Network Deployment when you install on a cluster, you can specify a different cluster for the push service. On Liberty collective, the push service is installed in a Liberty member that can be the same as the one where the runtime is installed.
- All the components use the same database system and the user. For DB2 , all the components also use the same schema.
- The Server Configuration Tool installs the components for a single server except for Liberty collective and WebSphere Application Server Network Deployment for asymmetric deployment. For an installation on multiple servers, a farm must be configured after the tool is run. The server farm configuration is not required on WebSphere Application Server Network Deployment.

For other topologies or other database settings, you can install the components with Ant Tasks or manually instead.

Running the Server Configuration Tool

Before you run the Server Configuration Tool, make sure that the following requirements are fulfilled:

- The databases and the tables for the components are prepared and ready to use. See Setting up databases (../databases).
- The server topology to install the components is decided. See Topologies and network flows (../topologies).
- The application server is configured. See Application server prerequisites.
- The user that runs the tool has the specific file system privileges. See File system prerequisites.

Click for instructions on running the Configuration Tool

After the installation is completed successfully, restart the application server in the case of Apache Tomcat or Liberty profile.

If Apache Tomcat is launched as a service, the setenv.bat or setenv.sh file that contains the statement to open the RMI might not be read. As a result, MobileFirst Server might not be able to work correctly. To set the required variables, see Configuring JMX connection for Apache Tomcat.

On WebSphere Application Server Network Deployment, the applications are installed but not started. You need to start them manually. You can do that from the WebSphere Application Server administration console.

Keep the configuration file in the Server Configuration Tool. You might reuse it to install the interim fixes. The menu to apply an interim fix is **Configurations > Replace the deployed WAR files**.

Applying a fix pack by using the Server Configuration Tool

If MobileFirst Server is installed with the configuration tool and the configuration file is kept, you can apply a fix pack or an interim fix by reusing the configuration file.

1. Start the Server Configuration Tool.
    - On Linux, from application shortcuts **Applications → IBM MobileFirst Platform Server → Server Configuration Tool** .
    - On Windows, click **Start → Programs → IBM MobileFirst Platform Server → Server Configuration Tool** .
    - On macOS, open a shell console. Go to **mfp_server_install_dir/shortcuts** and type **./configuration-tool.sh**.
    - The **mfp_server_install_dir** directory is where you installed MobileFirst Server.
2. Click **Configurations → Replace the deployed WAR files** and select an existing configuration to apply the fix pack or an interim fix.

Installing with Ant tasks

Use Ant tasks to install the MobileFirst Server components to your application server.

You can find the sample configuration files for installing MobileFirst Server in the **mfp_install_dir/MobileFirstServer/configuration-samples directory**.

You can also create a configuration with the Server Configuration Tool and export the Ant files by using **File → Export Configuration as Ant Files...**. The sample Ant files have the same limitations as the Server Configuration Tool:

- All components (MobileFirst Operations Console, MobileFirst Server administration service, MobileFirst Server live update service, the MobileFirst Server artifacts, and MobileFirst runtime) are in the same application server. However, on WebSphere Application Server Network Deployment when you install on a cluster, you can specify a different cluster for the administration and live update services, and for the runtime.
- If the MobileFirst Server push service is installed, it is also installed on the same server. However, on WebSphere Application Server Network Deployment when you install on a cluster, you can specify a different cluster for the push service.
- All the components use the same database system and the user. For DB2 , all the components also use the same schema.
- The Server Configuration Tool installs the components for a single server. For an installation on multiple servers, a farm must be configured after the tool is run. The server farm configuration is not supported on WebSphere Application Server Network Deployment.

You can configure the MobileFirst Server services to run in server farm with Ant tasks. To include your server in a farm, you need to specify some specific attributes that configure your application server accordingly. For more information about configuring a server farm with Ant tasks, see Installing a server farm with Ant tasks.

For other topologies that are supported in Topologies and network flows (../topologies), you can modify the sample Ant files.

The references to the Ant tasks are as follows:

- Ant tasks for installation of MobileFirst Operations Console, MobileFirst Server artifacts, MobileFirst Server administration, and live update services (../installation-reference/#ant-tasks-for-installation-of-mobilefirst-operations-console-mobilefirst-server-artifacts-mobilefirst-server-administration-and-live-update-services)
- Ant tasks for installation of MobileFirst Server push service (../installation-reference/#ant-tasks-for-installation-of-mobilefirst-server-push-service)
- Ant tasks for installation of MobileFirst runtime environments (../installation-reference/#ant-tasks-for-installation-of-mobilefirst-runtime-environments)

For an overview of installing with the sample configuration file and tasks, see Installing MobileFirst Server in command line mode (../tutorials/command-line).

You can run an Ant file with the Ant distribution that is part of the product installation. For example, if you have WebSphere Application Server Network Deployment cluster and your database is IBM DB2, you can use the **mfp_install_dir/MobileFirstServer/configuration-samples/configure-wasnd-cluster-db2.xml** Ant file. After you edit the file and enter all the required properties, you can run the following commands from **mfp_install_dir/MobileFirstServer/configuration-samples** directory:

- **mfp_install_dir/shortcuts/ant -f configure-wasnd-cluster-db2.xml help** - This command displays the list of all the possible targets of the Ant file, to install, uninstall, or update some components.
- **mfp_install_dir/shortcuts/ant -f configure-wasnd-cluster-db2.xml install** - This command installs MobileFirst Server on the WebSphere Application Server Network Deployment cluster, with DB2 as a data source by using the parameters that you entered in the properties of the Ant file.

After the installation, make a copy of the Ant file so that you can reuse it to apply a fix pack.

Applying a fix pack by using the Ant files

Updating with the sample Ant file

If you use the sample Ant files that are provided in the **mfp_install_dir/MobileFirstServer/configuration-samples** directory to install MobileFirst Server, you can reuse a copy of this Ant file to apply a fix pack. For password values, you can enter 12 stars (*) instead of the actual value, to be prompted interactively when the Ant file is run.

1. Verify the value of the **mfp.server.install.dir** property in the Ant file. It must point to the directory that contains the product with the fix pack applied. This value is used to take the updated MobileFirst Server WAR files.
2. Run the command: `mfp_install_dir/shortcuts/ant -f your_ant_file update`

Updating with own Ant file

If you use your own Ant file, make sure that for each installation task (**installmobilefirstadmin**, **installmobilefirstruntime**, and **installmobilefirstpush**), you have a corresponding update task in your Ant file with the same parameters. The corresponding update tasks are **updatemobilefirstadmin**, **updatemobilefirstruntime**, and **updatemobilefirstpush**.

1. Verify the class path of the **taskdef** element for the **mfp-ant-deployer.jar** file. It must point to the **mfp-ant-deployer.jar** file in an MobileFirst Server installation that the fix pack is applied. By default, the updated MobileFirst Server WAR files are taken from the location of **mfp-ant-deployer.jar**.
2. Run the update tasks (**updatemobilefirstadmin**, **updatemobilefirstruntime**, and **updatemobilefirstpush**) of your Ant file.

Sample Ant files modifications

You can modify the sample Ant files that are provided in the **mfp_install_dir/MobileFirstServer/configuration-samples** directory to adapt to your installation requirements.
The following sections provide the details on how you can modify the sample Ant files to adapt the installation to your needs:

1. Specify extra JNDI properties
2. Specify existing users
3. Specify Liberty Java EE level
4. Specify data source JDBC properties
5. Run the Ant files on a computer where MobileFirst Server is not installed
6. Specify WebSphere Application Server Network Deployment targets
7. Manual configuration of the RMI port on Apache Tomcat

Specify extra JNDI properties

The **installmobilefirstadmin**, **installmobilefirstruntime**, and **installmobilefirstpush** Ant tasks declare the values for the JNDI properties that are required for the components to function. These JNDI properties are used to define the JMX communication, and also the links to other components (such the live update service, the push service, the analytics service, or the authorization server). However, you can also define values for other JNDI properties. Use the `<property>` element that exists for these three tasks. For a list of JNDI properties, see:

- List of JNDI properties for MobileFirst Server administration service (../server-configuration/#list-of-jndi-properties-for-mobilefirst-server-administration-service)
- List of JNDI properties for MobileFirst Server push service (../server-configuration/#list-of-jndi-properties-for-mobilefirst-server-push-service)
- List of JNDI properties for MobileFirst runtime (../server-configuration/#list-of-jndi-properties-for-mobilefirst-runtime)

For example:

```
<installmobilefirstadmin ..>
   <property name="mfp.admin.actions.prepareTimeout" value="3000000"/>
</installmobilefirstadmin>
```

Specify existing users

By default, the **installmobilefirstadmin** Ant task creates users:

- On WebSphere Application Server Liberty to define a Liberty administrator for the JMX communication.
- On any application server, to define a user that is used for the communication with the live update service.

To use an existing user instead of creating new user, you can do the following operations:

1. In the `<jmx>` element, specify a user and password, and set the value of the **createLibertyAdmin** attribute to false. For example:

   ```
   <installmobilefirstadmin ...>
      <jmx libertyAdminUser="myUser" libertyAdminPassword="password" createLibertyAdmin="false" />
      ...
   ```

2. In the `<configuration>` element, specify a user and password and set the value of the **createConfigAdminUser** attribute to false. For example:

```
<installmobilefirstadmin ...>
  <configuration configAdminUser="myUser" configAdminPassword="password" createConfigAdminUser="false" />
  ...
```

Also, the user that is created by the sample Ant files is mapped to the security roles of the administration service and the console. With this setting, you can use this user to log on to MobileFirst Server after the installation. To change that behavior, remove the `<user>` element from the sample Ant files. Alternatively, you can remove the **password** attribute from the `<user>` element, and the user is not created in the local registry of the application server.

Specify Liberty Java EE level

Some distributions of WebSphere Application Server Liberty support features from Java EE 6 or from Java EE 7. By default, the Ant tasks automatically detect the features to install. For example, **jdbc-4.0** Liberty feature is installed for Java EE 6 and **jdbc-4.1** feature is installed in case of Java EE 7. If the Liberty installation supports both features from Java EE 6 and Java EE 7, you might want to force a certain level of features. An example might be that you plan to run both MobileFirst Server V8.0.0 and V7.1.0 on the same Liberty server. MobileFirst ServerV7.1.0 or earlier supports only Java EE 6 features.

To force a certain level of Java EE 6 features, use the jeeversion attribute of the `<websphereapplicationserver>` element. For example:

```
<installmobilefirstadmin execute="${mfp.process.admin}" contextroot="${mfp.admin.contextroot}">
  [...]
  <applicationserver>
    <websphereapplicationserver installdir="${appserver.was.installdir}"
      profile="Liberty" jeeversion="6">
```

Specify data source JDBC properties

You can specify the properties for the JDBC connection. Use the `<property>` element of a `<database>` element. The element is available in **configureDatabase**, **installmobilefirstadmin**, **installmobilefirstruntime**, and **installmobilefirstpush** Ant tasks. For example:

```
<configuredatabase kind="MobileFirstAdmin">
  <db2 database="${database.db2.mfpadmin.dbname}"
    server="${database.db2.host}"
    instance="${database.db2.instance}"
    user="${database.db2.mfpadmin.username}"
    port= "${database.db2.port}"
    schema = "${database.db2.mfpadmin.schema}"
    password="${database.db2.mfpadmin.password}">

    <property name="commandTimeout" value="10"/>
  </db2>
```

Run the Ant files on a computer where MobileFirst Server is not installed

To run the Ant tasks on a computer where MobileFirst Server is not installed, you need the following items:

- An Ant installation
- A copy of the **mfp-ant-deployer.jar** file to the remote computer. This library contains the definition of the Ant tasks.
- To specify the resources to be installed. By default, the WAR files are taken near the **mfp-ant-deployer.jar**, but you can specify the location of these WAR files. For example:

```
<installmobilefirstadmin execute="true" contextroot="/mfpadmin" serviceWAR="/usr/mfp/mfp-admin-service.war">
  <console install="true" warFile="/usr/mfp/mfp-admin-ui.war"/>
```

For more information, see the Ant tasks to install each MobileFirst Server component at Installation reference (../installation-reference).

Specify WebSphere Application Server Network Deployment targets

To install on WebSphere Application Server Network Deployment, the specified WebSphere Application Server profile must be the deployment manager. You can deploy on the following configurations:

- A cluster
- A single server
- A cell (all the servers of a cell)
- A node (all the servers of a node)

The sample files such as **configure-wasnd-cluster-dbms-name.xml**, **configure-wasnd-server-dbms-name.xml**, and **configure-wasnd-node-dbms-name.xml** contain the declaration to deploy on each type of target. For more information, see the Ant tasks to install each MobileFirst Server component in the Installation reference (../installation-reference).

> Note: As of V8.0.0, the sample configuration file for the WebSphere Application Server Network Deployment cell is not provided.

Manual configuration of the RMI port on Apache Tomcat

By default, the Ant tasks modify the **setenv.bat** file or the **setenv.sh** file to open the RMI port. If you prefer to open the RMI port manually, add the **tomcatSetEnvConfig** attribute with the value as false to the `<jmx>` element of the **installmobilefirstadmin**, **updatemobilefirstadmin**, and **uninstallmobilefirstadmin** tasks.

Installing the MobileFirst Server components manually

You can also install the MobileFirst Server components to your application server manually.
The following topics provide you the complete information to guide you through the installing process of the components on the supported applications in production.

- Manual installation on WebSphere Application Server Liberty
- Manual installation on WebSphere Application Server Liberty collective
- Manual installation on Apache Tomcat
- Manual installation on WebSphere Application Server and WebSphere Application Server Network Deployment

Manual installation on WebSphere Application Server Liberty

Make sure that you have also fulfilled the requirements as documented in WebSphere Application Server Liberty prerequisites.

- Topology constraints
- Application server settings
- Liberty features required by the MobileFirst Server applications
- Global JNDI entries
- Class loader
- Password decoder user feature
- Configuration details

Topology constraints

The MobileFirst Server administration service, the MobileFirst Server live update service, and the MobileFirst runtime must be installed on the same application server. The context root of the live update service must be defined as **the-adminContextRootconfig**. The context root of the push service must be **imfpush**. For more information about the constraints, see Constraints on the MobileFirst Server components and MobileFirst Analytics (../topologies/#constraints-on-the-mobilefirst-server-components-and-mobilefirst-analytics).

Application server settings

You must configure the **webContainer** element to load the servlets immediately. This setting is required for the initialization through JMX. For example: `<webContainer deferServletLoad="false"/>`.

Optionally, to avoid timeout issues that break the startup sequence of the runtime and the administration service on some Liberty versions, change the default **executor** element. Set large values to the **coreThreads** and **maxThreads** attributes. For example:

```
<executor id="default" name="LargeThreadPool"
  coreThreads="200" maxThreads="400" keepAlive="60s"
  stealPolicy="STRICT" rejectedWorkPolicy="CALLER_RUNS"/>
```

You might also configure the **tcpOptions** element and set the **soReuseAddr** attribute to `true`: `<tcpOptions soReuseAddr="true"/>`.

Liberty features required by the MobileFirst Server applications

You can use the following features for Java EE 6 or Java EE 7.

MobileFirst Server administration service

- **jdbc-4.0** (jdbc-4.1 for Java EE 7)
- **appSecurity-2.0**
- **restConnector-1.0**
- **usr:MFPDecoderFeature-1.0**

MobileFirst Server push service

- **jdbc-4.0** (jdbc-4.1 for Java EE 7)
- **servlet-3.0** (servlet-3.1 for Java EE 7)

- **ssl-1.0**
- **usr:MFPDecoderFeature-1.0**

MobileFirst runtime

- **jdbc-4.0** (jdbc-4.1 for Java EE 7)
- **servlet-3.0** (servlet-3.1 for Java EE 7)
- **ssl-1.0**
- **usr:MFPDecoderFeature-1.0**

Global JNDI entries

The following global JNDI entries are required to configure the JMX communication between the runtime and the administration service:

- **mfp.admin.jmx.host**
- **mfp.admin.jmx.port**
- **mfp.admin.jmx.user**
- **mfp.admin.jmx.pwd**
- **mfp.topology.platform**
- **mfp.topology.clustermode**

These global JNDI entries are set with this syntax and are not prefixed by a context root. For example: `<jndiEntry jndiName="mfp.admin.jmx.port" value="9443"/>`.

> **Note:** To protect against an automatic conversion of the JNDI values, so that 075 is not converted to 61 or 31.500 is not converted to 31.5, use this syntax '"075"' when you define the value.

For more information about the JNDI properties for the administration service, see List of JNDI properties for MobileFirst Server administration service (../server-configuration/#list-of-jndi-properties-for-mobilefirst-server-administration-service).

For a farm configuration, see also the following topics:

- Server farm topology (../topologies/#server-farm-topology)
- Topologies and network flows (../topologies)
- Installing a server farm

Class loader

For all applications, the class loader must have the parent last delegation. For example:

```
<application id="mfpadmin" name="mfpadmin" location="mfp-admin-service.war" type="war">
  [...]
  <classloader delegation="parentLast">
  </classloader>
</application>
```

Password decoder user feature

Copy the password decoder user feature to your Liberty profile. For example:

- On UNIX and Linux systems:

  ```
  mkdir -p LIBERTY_HOME/wlp/usr/extension/lib/features
  cp product_install_dir/features/com.ibm.websphere.crypto_1.0.0.jar LIBERTY_HOME/wlp/usr/extension/lib/
  cp product_install_dir/features/MFPDecoderFeature-1.0.mf LIBERTY_HOME/wlp/usr/extension/lib/features/
  ```

- On Windows systems:

  ```
  mkdir LIBERTY_HOME\wlp\usr\extension\lib
  copy /B product_install_dir\features\com.ibm.websphere.crypto_1.0.0.jar
  LIBERTY_HOME\wlp\usr\extension\lib\com.ibm.websphere.crypto_1.0.0.jar
  mkdir LIBERTY_HOME\wlp\usr\extension\lib\features
  copy /B product_install_dir\features\MFPDecoderFeature-1.0.mf
  LIBERTY_HOME\wlp\usr\extension\lib\features\MFPDecoderFeature-1.0.mf
  ```

Configuration details

MobileFirst Server administration service configuration details

MobileFirst Server live update service configuration details

MobileFirst Operations Console configuration details

MobileFirst runtime configuration details

MobileFirst Server push service configuration details

MobileFirst Server artifacts configuration details

Manual installation on WebSphere Application Server Liberty collective

Make sure that you have also fulfilled the requirements as documented in WebSphere Application Server Liberty prerequisites.

- Topology constraints
- Application server settings
- Liberty features required by the MobileFirst Server applications
- Global JNDI entries
- Class loader
- Password decoder user feature
- Configuration details

Topology constraints

The MobileFirst Server administration service, the MobileFirst Server live update service, and MobileFirst Operations Console must be installed in a Liberty collective controller. The MobileFirst runtime and the MobileFirst Server push service must be installed in every member of the Liberty collective cluster.

The context root of the live update service must be defined as **the-adminContextRootconfig**. The context root of the push service must be **imfpush**. For more information about the constraints, see Constraints on the MobileFirst Server components and MobileFirst Analytics (../topologies/#constraints-on-the-mobilefirst-server-components-and-mobilefirst-analytics).

Application server settings

You must configure the **webContainer** element to load the servlets immediately. This setting is required for the initialization through JMX. For example: `<webContainer deferServletLoad="false"/>`.

Optionally, to avoid timeout issues that break the startup sequence of the runtime and the administration service on some Liberty versions, change the default **executor** element. Set large values to the **coreThreads** and **maxThreads** attributes. For example:

```
<executor id="default" name="LargeThreadPool"
  coreThreads="200" maxThreads="400" keepAlive="60s"
  stealPolicy="STRICT" rejectedWorkPolicy="CALLER_RUNS"/>
```

You might also configure the **tcpOptions** element and set the **soReuseAddr** attribute to `true`: `<tcpOptions soReuseAddr="true"/>`.

Liberty features required by the MobileFirst Server applications

You need to add the following features for Java EE 6 or Java EE 7.

MobileFirst Server administration service

- **jdbc-4.0** (jdbc-4.1 for Java EE 7)
- **appSecurity-2.0**
- **restConnector-1.0**
- **usr:MFPDecoderFeature-1.0**

MobileFirst Server push service

- **jdbc-4.0** (jdbc-4.1 for Java EE 7)
- **servlet-3.0** (servlet-3.1 for Java EE 7)
- **ssl-1.0**
- **usr:MFPDecoderFeature-1.0**

MobileFirst runtime

- **jdbc-4.0** (jdbc-4.1 for Java EE 7)
- **servlet-3.0** (servlet-3.1 for Java EE 7)
- **ssl-1.0**
- **usr:MFPDecoderFeature-1.0**

Global JNDI entries

The following global JNDI entries are required to configure the JMX communication between the runtime and the administration service:

- **mfp.admin.jmx.host**
- **mfp.admin.jmx.port**
- **mfp.admin.jmx.user**
- **mfp.admin.jmx.pwd**
- **mfp.topology.platform**
- **mfp.topology.clustermode**
- **mfp.admin.serverid**

These global JNDI entries are set with this syntax and are not prefixed by a context root. For example: `<jndiEntry jndiName="mfp.admin.jmx.port" value="9443"/>`.

> **Note:** To protect against an automatic conversion of the JNDI values, so that 075 is not converted to 61 or 31.500 is not converted to 31.5, use this syntax '"075"' when you define the value.

- For more information about the JNDI properties for the administration service, see List of JNDI properties for MobileFirst Server administration service (../server-configuration/#list-of-jndi-properties-for-mobilefirst-server-administration-service).
- For more information about the JNDI properties for the runtime, see List of JNDI properties for MobileFirst runtime (../server-configuration/#list-of-jndi-properties-for-mobilefirst-runtime).

Class loader

For all applications, the class loader must have the parent last delegation. For example:

```
<application id="mfpadmin" name="mfpadmin" location="mfp-admin-service.war" type="war">
  [...]
  <classloader delegation="parentLast">
  </classloader>
</application>
```

Password decoder user feature

Copy the password decoder user feature to your Liberty profile. For example:

- On UNIX and Linux systems:

  ```
  mkdir -p LIBERTY_HOME/wlp/usr/extension/lib/features
  cp product_install_dir/features/com.ibm.websphere.crypto_1.0.0.jar LIBERTY_HOME/wlp/usr/extension/lib/
  cp product_install_dir/features/MFPDecoderFeature-1.0.mf LIBERTY_HOME/wlp/usr/extension/lib/features/
  ```

- On Windows systems:

  ```
  mkdir LIBERTY_HOME\wlp\usr\extension\lib
  copy /B product_install_dir\features\com.ibm.websphere.crypto_1.0.0.jar
  LIBERTY_HOME\wlp\usr\extension\lib\com.ibm.websphere.crypto_1.0.0.jar
  mkdir LIBERTY_HOME\wlp\usr\extension\lib\features
  copy /B product_install_dir\features\MFPDecoderFeature-1.0.mf
  LIBERTY_HOME\wlp\usr\extension\lib\features\MFPDecoderFeature-1.0.mf
  ```

Configuration details

MobileFirst Server administration service configuration details

MobileFirst Server live update service configuration details

MobileFirst Operations Console configuration details

MobileFirst runtime configuration details

MobileFirst Server push service configuration details

MobileFirst Server artifacts configuration details

Manual installation on Apache Tomcat

Make sure that you have fulfilled the requirements as documented in Apache Tomcat prerequisites.

- Topology constraints
- Application server settings

- Configuration details

Topology constraints

The MobileFirst Server administration service, the MobileFirst Server live update service, and the MobileFirst runtime must be installed on the same application server. The context root of the live update service must be defined as **the-adminContextRootconfig**. The context root of the push service must be **imfpush**. For more information about the constraints, see Constraints on the MobileFirst Server components and MobileFirst Analytics (../topologies/#constraints-on-the-mobilefirst-server-components-and-mobilefirst-analytics).

Application server settings

ou must activate the **Single Sign On Valve**. For example:

```
<Valve className="org.apache.catalina.authenticator.SingleSignOn"/>
```

Optionally, you might want to activate the memory realm if the users are defined in **tomcat-users.xml**. For example:

```
<Realm className="org.apache.catalina.realm.MemoryRealm"/>
```

Configuration details

MobileFirst Server administration service configuration details

MobileFirst Server live update service configuration details

MobileFirst Operations Console configuration details

MobileFirst runtime configuration details

MobileFirst Server push service configuration details

MobileFirst Server artifacts configuration details

Manual installation on WebSphere Application Server and WebSphere Application Server Network Deployment

Make sure that you have fulfilled the requirements as documented in WebSphere Application Server and WebSphere Application Server Network Deployment prerequisites.

- Topology constraints
- Application server settings
- Class loader
- Configuration details

Topology constraints

On a stand-alone WebSphere Application Server
The MobileFirst Server administration service, the MobileFirst Server live update service, and the MobileFirst runtime must be installed on the same application server. The context root of the live update service must be defined as **the-adminContextRootConfig**. The context root of the push service must be **imfpush**. For more information about the constraints, see Constraints on the MobileFirst Server components and MobileFirst Analytics (../topologies/#constraints-on-the-mobilefirst-server-components-and-mobilefirst-analytics).

On WebSphere Application Server Network Deployment
The deployment manager must be running while MobileFirst Server is running. The deployment manager is used for the JMX communication between the runtime and the administration service. The administration service and the live update service must be installed on the same application server. The runtime can be installed on different servers than the administration service, but it must be on the same cell.

Application server settings

The administrative security and the application security must be enabled. You can enable the application security in the WebSphere Application Server administration console:

1. Log in to the WebSphere Application Server administration console.
2. Click **Security → Global Security**. Ensure that Enable administrative security is selected.
3. Also, ensure that **Enable application security** is selected. The application security can be enabled only if administrative security is enabled.
4. Click **OK**.
5. Save the changes.

For more information, see Enabling security (http://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/tsec_csec2.html?view=kc) in WebSphere Application Server documentation.

The server class loader policy must support parent last delegation. The MobileFirst Server WAR files must be installed with parent last class loader mode. Review the class-loader policy:

1. Log in to the WebSphere Application Server administration console.
2. Click S**ervers → Server Types → WebSphere application servers**, and click on the server that is used for IBM MobileFirst Foundation.
3. If the class-loader policy is set to **Multiple**, do nothing.
4. If the class-loader policy is set to **Single** and the class loading mode is set to **Classes loaded with local class loader first (parent last)**, do nothing.
5. If the class-loader policy is set to **Single** and the class loading mode is set to **Classes loaded with parent class loader first (parent first)**, change the class-loader policy to **Multiple**. Also, set the class loader order of all applications other than MobileFirst Server applications to **Classes loaded with parent class loader first (parent first)**.

Class loader

For all MobileFirst Server applications, the class loader must have the parent last delegation.

To set the class loader delegation to parent last after an application is installed, follow these steps:

1. Click the **Manage Applications** link, or click **Applications → Application Types → WebSphere entreprise applications**.
2. Click the **MobileFirst Server** application. By default the name of the application is the name of the WAR file.
3. In the **Detail Properties** section, click the **Class loading and update detection** link.
4. In the **Class loader order** pane, select the **Classes loaded with local class loader first (parent last)** option.
5. Click **OK**.
6. In the **Modules** section, click the **Manage Modules** link.
7. Click the module.
8. For the **Class loader order** field, select the **Classes loaded with local class loader first (parent last)** option.
9. Click **OK** twice to confirm the selection and back to the **Configuration** panel of the application.
10. Click **Save** to persist the changes.

Configuration details

MobileFirst Server administration service configuration details

MobileFirst Server live update service configuration details

MobileFirst Operations Console configuration details

MobileFirst runtime configuration details

MobileFirst Server push service configuration details

MobileFirst Server artifacts configuration details

Installing a server farm

You can install your server farm by running Ant tasks, with the Server Configuration Tool, or manually.

- Planning the configuration of a server farm
- Installing a server farm with the Server Configuration Tool
- Installing a server farm with Ant tasks
- Configuring a server farm manually
- Verifying a farm configuration
- Lifecycle of a server farm node

Planning the configuration of a server farm

To plan the configuration of a server farm, choose the application server, configure the MobileFirst databases, and deploy the WAR files of the MobileFirst Server components on each server of the farm. You have the options to use the Server Configuration Tool, Ant tasks, or manual operations to configure a server farm.

When you intend to plan a server farm installation, 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) first, and in particular see Server farm topology (../topologies/#server-farm-topology).

In IBM MobileFirst Foundation, a server farm is composed of multiple stand-alone application servers that are not federated or administered by a managing component of an application server. MobileFirst Server internally provides a farm plug-in as the means to enhance an application server so that it can be part of a server farm.

When to declare a server farm

Declare a server farm in the following cases:

- MobileFirst Server is installed on multiple Tomcat application servers.
- MobileFirst Server is installed on multiple WebSphere Application Server servers but not on WebSphere Application Server Network Deployment.
- MobileFirst Server is installed on multiple WebSphere Application Server Liberty servers.

Do not declare a server farm in the following cases:

- Your application server is stand-alone.
- Multiple application servers are federated by WebSphere Application Server Network Deployment.

Why it is mandatory to declare a farm

Each time a management operation is performed through MobileFirst Operations Console or through the MobileFirst Server administration service application, the operation needs to be replicated to all instances of a runtime environment. Examples of such management operations are the uploading of a new version of an app or of an adapter. The replication is done via JMX calls performed by the administration service application instance that handles the operation. The administration service needs to contact all runtime instances in the cluster. In environments listed under **When to declare a server farm** above, the runtime can be contacted through JMX only if a farm is configured. If a server is added to a cluster without proper configuration of the farm, the runtime in that server will be in an inconsistent state after each management operation, and until it is restarted again.

Installing a server farm with the Server Configuration Tool

Use the Server Configuration Tool to configure each server in the farm according to the requirements of the single type of application server that is used for each member of the server farm.

When you plan a server farm with the Server Configuration Tool, first create the stand-alone servers and configure their respective truststores so that they can communicate with one another in a secure way. Then, run the tool that does the following operations:

- Configure the database instance that is shared by the MobileFirst Server components.
- Deploy the MobileFirst Server components to each server
- Modify its configuration to make it a member of a server farm

Click for instructions on installing a server farm with the Server Configuration Tool

Installing a server farm with Ant tasks

Use Ant tasks to configure each server in the farm according to the requirements of the single type of application server that is used for each member of the server farm.

When you plan a server farm with Ant tasks, first create the stand-alone servers and configure their respective truststores so that they can communicate with one another in a secure way. Then, run Ant tasks to configure the database instance that is shared by the MobileFirst Server components. Finally, run Ant tasks to deploy the MobileFirst Server components to each server and modify its configuration to make it a member of a server farm.

Click for instructions on installing a server farm with Ant tasks

Configuring a server farm manually

You must configure each server in the farm according to the requirements of the single type of application server that is used for each member of the server farm.

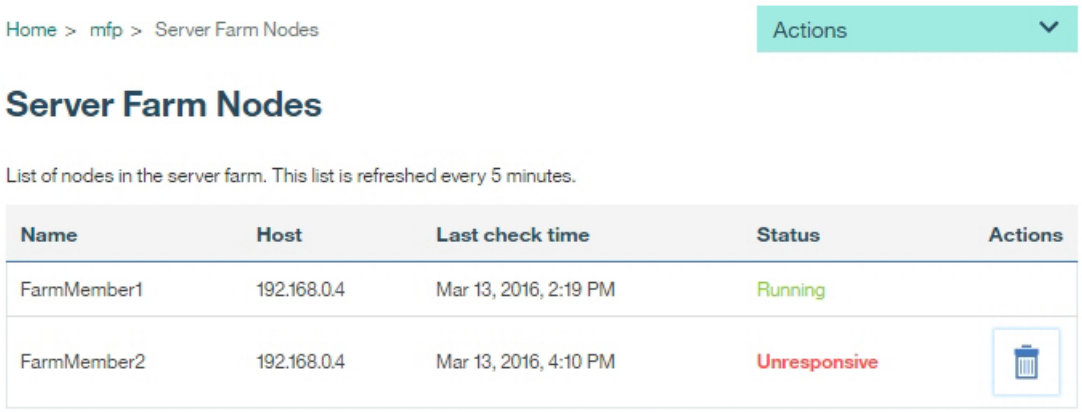
When you plan a server farm, first create stand-alone servers that communicate with the same database instance. Then, modify the configuration of these servers to make them members of a server farm.

Click for instructions on configuring a server farm manually

Verifying a farm configuration

The purpose of this task is to check the status of the farm members and verify whether a farm is configured properly.

1. Start all the servers of the farm.
2. Access MobileFirst Operations Console. For example, **http://server_name:port/mfpconsole**, or **https://hostname:secure_port/mfpconsole** in HTTPS. In the console sidebar, an extra menu that is labeled as Server Farm Nodes appears.
3. Click **Server Farm Nodes** to access the list of registered farm members and their status. In the following example, the node that is identified as **FarmMember2** is considered to be down, which indicates that this server has probably failed and requires some maintenance.



Lifecycle of a server farm node

You can configure heartbeat rate and timeout values to indicate possible server problems among farm members by triggering a change in status of an affected node.

Registration and monitoring servers as farm nodes

When a server configured as a farm node is started, the administration service on that server automatically registers it as a new farm member. When a farm member is shut down, it automatically unregisters from the farm.

A heartbeat mechanism exists to keep track of farm members that might become unresponsive, for example, because of a power outage or a server failure. In this heartbeat mechanism, MobileFirst runtimes periodically send a heartbeat to MobileFirst administration services at a specified rate. If the MobileFirst administration service registers that too long a time has elapsed since a farm member sent a heartbeat, the farm member is considered to be down.

Farm members that are considered to be down do not serve any more requests to mobile applications.

Having one or more nodes down does not prevent the other farm members from correctly serving requests to mobile applications nor from accepting new management operations that are triggered through the MobileFirst Operations Console.

Configuring the heartbeat rate and timeout values

You can configure the heartbeat rate and timeout values by defining the following JNDI properties:

- **mfp.admin.farm.heartbeat**
- **mfp.admin.farm.missed.heartbeats.timeout**

For more information about JNDI properties, see List of JNDI properties for MobileFirst Server administration service (../server-configuration/#list-of-jndi-properties-for-mobilefirst-server-administration-service).

Last modified on