Configuring MobileFirst Server

Overview

Consider your backup and recovery policy, optimize your MobileFirst Server configuration, and apply access restrictions and security options.

Jump to

Endpoints of the MobileFirst Server production server

You can create whitelists and blacklists for the endpoints of the IBM MobileFirst Server.

> **Note:** Information regarding URLs that are exposed by IBM MobileFirst Foundation is provided as a guideline. Organizations must ensure the URLs are tested in an enterprise infrastructure, based on what has been enabled for white and black lists.

API URL under `<runtime context root>/api/`	**Description**	**Suggested for whitelist?**
/adapterdoc/*	Return the adapter's Swagger documentation for the named adapter	No. Used only internally by the administrator and the developers
/adapters/*	Adapters serving	Yes
/az/v1/authorization/*	Authorize the client to access a specific scope	Yes
/az/v1/introspection	Introspect the client's access token	No. This API is for confidential clients only.
/az/v1/token	Generate an access token for the client	Yes
/clientLogProfile/*	Get client log profile	Yes
/directupdate/*	Get Direct Update .zip file	Yes, if you plan to use Direct Update
/loguploader	Upload client logs to server	Yes
/preauth/v1/heartbeat	Accept heartbeat from the client and note the last activity time	Yes
/preauth/v1/logout	Log out from a security check	Yes
/preauth/v1/preauthorize	Map and execute security checks for a specific scope	Yes
/reach	The server is reachable	No, for internal use only
/registration/v1/clients/*	Registration-service clients API	No. This API is for confidential clients only.
/registration/v1/self/*	Registration-service client self-registration API	Yes

Configuring MobileFirst Server to enable TLS V1.2

For MobileFirst Server to communicate with devices that support only Transport Layer Security v1.2 (TLS) V1.2, among the SSL protocols, you must complete the following instructions.

The steps to configure MobileFirst Server to enable Transport Layer Security (TLS) V1.2 depend on how MobileFirst Server connects to devices.

- If MobileFirst Server is behind a reverse proxy that decrypts SSL-encoded packets from devices before it passes the packets to the application server, you must enable TLS V1.2 support on your reverse proxy. If you use IBM HTTP Server as your reverse proxy, see Securing IBM HTTP Server (http://www.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.ihs.doc/ihs/welc6top_securing_ihs_container.html?view=kc) for instructions.
- If MobileFirst Server communicates directly with devices, the steps to enable TLS V1.2 depend on whether your application serveris Apache Tomcat, WebSphere Application Server Liberty profile, or WebSphere Application Server full profile.

Apache Tomcat

1. Confirm that the Java Runtime Environment (JRE) supports TLS V1.2. Ensure that you have one of the following JRE versions:
   - Oracle JRE 1.7.0_75 or later
   - Oracle JRE 1.8.0_31 or later
2. Edit the conf/server.xml file and modify the element that declares the HTTPS port so that the sslEnabledProtocols attribute has the following value: `sslEnabledProtocols="TLSv1.2,TLSv1.1,TLSv1,SSLv2Hello"`

WebSphere Application Server Liberty profile

1. Confirm that the Java Runtime Environment (JRE) supports TLS V1.2.
   - If you use an IBM Java SDK, ensure that your IBM Java SDK is patched for the POODLE vulnerability. You can find the minimum IBM Java SDK versions that contain the patch for your version of WebSphere Application Server in Security Bulletin: Vulnerability in SSLv3 affects IBM WebSphere Application Server (CVE-2014-3566) (http://www.ibm.com/support/docview.wss?uid=swg21687173).

     > **Note:** You can use the versions that are listed in the security bulletin or later versions.

   - If you use an Oracle Java SDK, ensure that you have one of the following versions:
     - Oracle JRE 1.7.0_75 or later
     - Oracle JRE 1.8.0_31 or later
2. If you use an IBM Java SDK, edit the **server.xml** file.
   - Add the following line: `<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore" sslProtocol="SSL_TLSv2"/>`
   - Add the `sslProtocol="SSL_TLSv2"` attribute to all existing `<ssl>` elements.

WebSphere Application Server full profile

1. Confirm that the Java Runtime Environment (JRE) supports TLS V1.2.
   Ensure that your IBM Java SDK is patched for the POODLE vulnerability. You can find the minimum IBM Java SDK versions that contain the patch for your version of WebSphere Application Server in Security Bulletin: Vulnerability in SSLv3 affects IBM WebSphere Application Server (CVE-2014-3566) (http://www.ibm.com/support/docview.wss?uid=swg21687173).

   > **Note:** You can use the versions that are listed in the security bulletin or later versions.

2. Log in to WebSphere Application Server administrative console, and click **Security → SSL certificate and key management → SSL configurations**.
3. For each SSL configuration listed, modify the configuration to enable TLS V1.2.
   - Select an SSL configuration and then, under **Additional Properties**, click **Quality of protections (QoP)** settings.
   - From the **Protocol** list, select **SSL_TLSv2**.
   - Click **Apply** and then save the changes.

Configuring user authentication for MobileFirst Server administration

MobileFirst Server administration requires user authentication. You can configure user authentication and choose an authentication method. Then, the configuration procedure depends on the web application server that you use.

> **Important:** If you use stand-alone WebSphere Application Server full profile, use an authentication method other than the simple WebSphere authentication method (SWAM) in global security. You can use lightweight third-party authentication (LTPA). If you use SWAM, you might experience unexpected authentication failures.

You must configure authentication after the installer deploys the MobileFirst Server administration web applications in the web application server.

The MobileFirst Server administration has the following Java Platform, Enterprise Edition (Java EE) security roles defined:

- mfpadmin
- mfpdeployer
- mfpoperator
- mfpmonitor

You must map the roles to the corresponding sets of users. The **mfpmonitor** role can view data but cannot change any data. The following tables list MobileFirst roles and functions for production servers.

Deployment

	Administrator	**Deployer**	**Operator**	**Monitor**
Java EE security role.	mfpadmin	mfpdeployer	mfpoperator	mfpmonitor
Deploy an application.	Yes	Yes	No	No
Deploy an adapter.	Yes	Yes	No	No

MobileFirst Server management

	Administrator	**Deployer**	**Operator**	**Monitor**
Java EE security role.	mfpadmin	mfpdeployer	mfpoperator	mfpmonitor
Configure runtime settings.	Yes	Yes	No	No

Application management

	Administrator	**Deployer**	**Operator**	**Monitor**
Java EE security role.	mfpadmin	mfpdeployer	mfpoperator	mfpmonitor
Upload new MobileFirst application.	Yes	Yes	No	No
Remove MobileFirst application.	Yes	Yes	No	No
Upload new MobileFirst adapter.	Yes	Yes	No	No
Remove MobileFirst adapter.	Yes	Yes	No	No
Turn on or off application authenticity testing for an application.	Yes	Yes	No	No
Change properties on MobileFirst application status: Active, Active Notifying, and Disabled.	Yes	Yes	Yes	No

Basically, all roles can issue GET requests, the **mfpadmin**, **mfpdeployer**, and **mfpmonitor** roles can also issue POST and PUT requests, and the **mfpadmin** and **mfpdeployer** roles can also issue DELETE requests.

Requests related to push notifications

	Administrator	**Deployer**	**Operator**	**Monitor**
Java EE security role.	mfpadmin	mfpdeployer	mfpoperator	mfpmonitor

	Administrator	Deployer	Operator	Monitor
GET requests • Get a list of all the devices that use push notification for an application • • Get the details of a specific device • Get the list of subscriptions • Get the subscription information that is associated with a subscription ID. • Get the details of a GCM configuration • Get the details of an APNS configuration • Get the list of tags that are defined for the application • Get details of a specific tag	Yes	Yes	Yes	Yes
POST and PUT requests • Register an app with push notification • Update a push device registration • Create a subscription • Add or update a GCM configuration • Add or update an APNS configuration • Submit notifications to a device • Create or update a tag	Yes	Yes	Yes	No
DELETE requests • Delete the registration of a device to push notification • Delete a subscription • Unsubscribe a device from a tag • Delete a GCM configuration • Delete an APNS configuration • Delete a tag	Yes	Yes	No	No

Disabling

	Administrator	Deployer	Operator	Monitor
Java EE security role.	mfpadmin	mfpdeployer	mfpoperator	mfpmonitor
Disable the specific device, marking the state as lost or stolen so that access from any of the applications on that device is blocked.	Yes	Yes	Yes	No
Disable a specific application, marking the state as disabled so that access from the specific application on that device is blocked.	Yes	Yes	Yes	No

If you choose to use an authentication method through a user repository such as LDAP, you can configure the MobileFirst Server administration so that you can use users and groups with the user repository to define the Access Control List (ACL) of the MobileFirst Server administration. This procedure depends on the type and version of the web application server that you use.

Configuring WebSphere Application Server full profile for MobileFirst Server administration

Configure security by mapping the MobileFirst Server administration Java EE roles to a set of users for both web applications.

You define the basics of user configuration in the WebSphere Application Server console. Access to the console is usually by this address: `https://localhost:9043/ibm/console/`

1. Select **Security → Global Security**.
2. Select **Security Configuration Wizard** to configure users. You can manage individual user accounts by selecting **Users and Groups → Manage Users**.
3. Map the roles **mfpadmin**, **mfpdeployer**, **mfpmonitor**, and **mfpoperator** to a set of users.
    - Select **Servers → Server Types → WebSphere application servers**.
    - Select the server.

- In the **Configuration** tab, select **Applications → Enterprise applications**.
- Select **MobileFirst*Administration*Service**.
- In the **Configuration** tab, select **Details → Security** role to user/group mapping.
- Perform the necessary customization.
- Click **OK**.
- Repeat the steps to map the roles for the console web application. This time select **MobileFirst*Administration*Console**.
- Click **Save** to save the changes.

Configuring WebSphere Application Server Liberty profile for MobileFirst Server administration

In WebSphere Application Server Liberty profile, you configure the roles of **mfpadmin**, **mfpdeployer**, **mfpmonitor**, and **mfpoperator** in the **server.xml** configuration file of the server.

To configure the security roles, you must edit the **server.xml** file. In the `<application-bnd>` element of each `<application>` element, create `<security-role>` elements. Each `<security-role>` element is for each roles: **mfpadmin**, mfpdeployer, mfpmonitor, and mfpoperator. Map the roles to the appropriate user group name, in this example: **mfpadmingroup**, **mfpdeployergroup**, **mfpmonitorgroup**, or **mfpoperatorgroup**. These groups are defined through the `<basicRegistry>` element. You can customize this element or replace it entirely with an `<ldapRegistry>` element or a `<safRegistry>` element.

Then, to maintain good response times with a large number of installed applications, for example with 80 applications, you should configure a connection pool for the administration database.

1. Edit the **server.xml** file. For example:

```
<security-role name="mfpadmin">
  <group name="mfpadmingroup"/>
</security-role>
<security-role name="mfpdeployer">
  <group name="mfpdeployergroup"/>
</security-role>
<security-role name="mfpmonitor">
  <group name="mfpmonitorgroup"/>
</security-role>
<security-role name="mfpoperator">
  <group name="mfpoperatorgroup"/>
</security-role>

<basicRegistry id="mfpadmin">
  <user name="admin" password="admin"/>
  <user name="guest" password="guest"/>
  <user name="demo" password="demo"/>
  <group name="mfpadmingroup">
    <member name="guest"/>
    <member name="demo"/>
  </group>
  <group name="mfpdeployergroup">
    <member name="admin" id="admin"/>
  </group>
  <group name="mfpmonitorgroup"/>
  <group name="mfpoperatorgroup"/>
</basicRegistry>
```

2. Define the **AppCenterPool** size:

```
<connectionManager id="AppCenterPool" minPoolSize="10" maxPoolSize="40"/>
```

3. In the `<dataSource>` element, define a reference to the connection manager:

```
<dataSource id="MFPADMIN" jndiName="mfpadmin/jdbc/mfpAdminDS" connectionManagerRef="AppCenterPool">
...
</dataSource>
```

Configuring Apache Tomcat for MobileFirst Server administration

You must configure the Java EE security roles for the MobileFirst Server administration on the Apache Tomcat web application server.

1. If you installed the MobileFirst Server administration manually, declare the following roles in the **conf/tomcat-users.xml** file:

```
<role rolename="mfpadmin"/>
<role rolename="mfpmonitor"/>
<role rolename="mfpdeployer"/>
<role rolename="mfpoperator"/>
```

2. Add roles to the selected users, for example:

```
<user name="admin" password="admin" roles="mfpadmin"/>
```

3. You can define the set of users as described in the Apache Tomcat documentation, Realm Configuration HOW-TO (http://tomcat.apache.org/tomcat-7.0-doc/realm-howto.html).

List of JNDI properties of the MobileFirst Server web applications

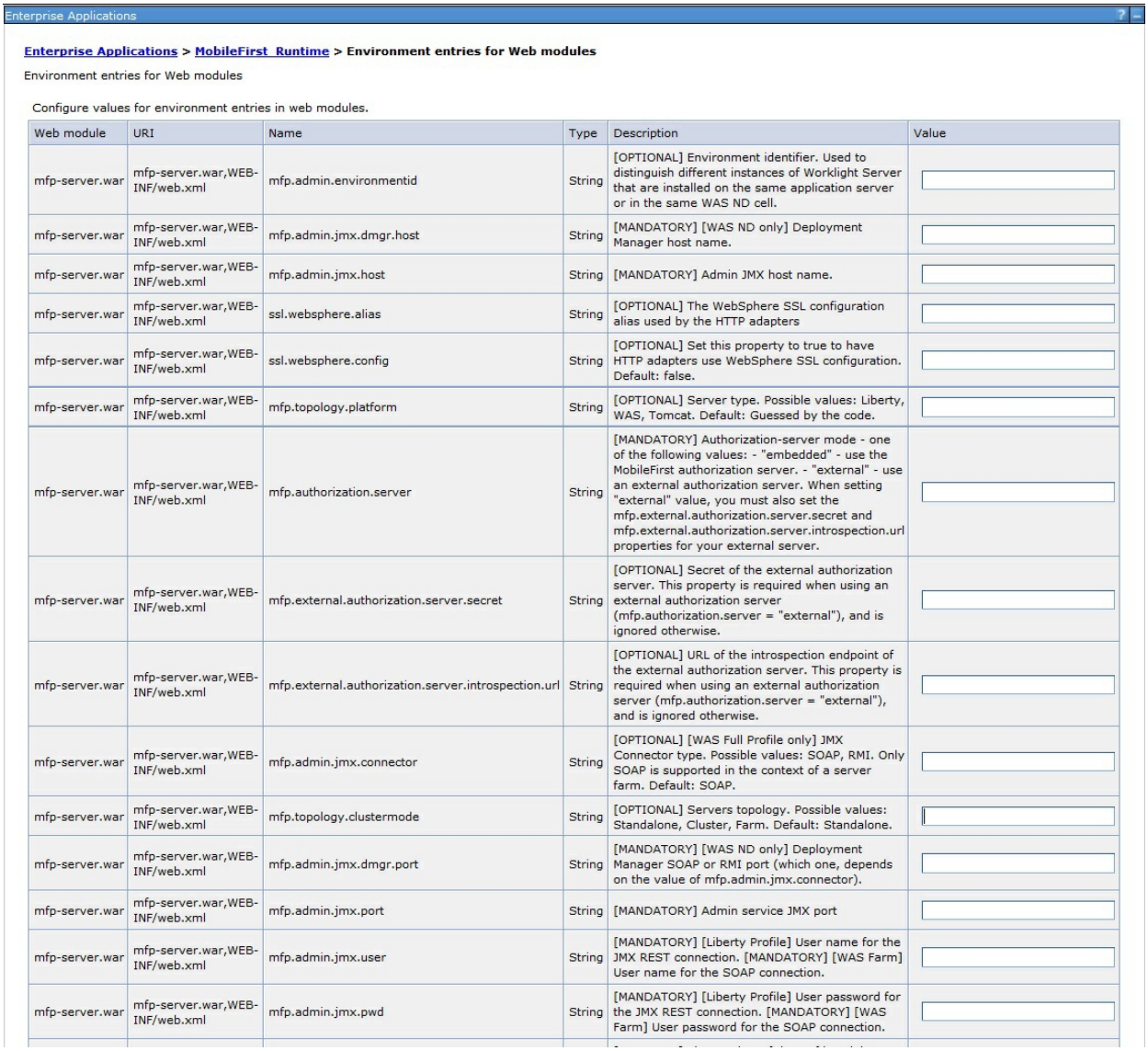
Configure the JNDI properties for the MobileFirst Server web applications that are deployed to the application server.

- Setting up JNDI properties for MobileFirst Server web applications
- List of JNDI properties for MobileFirst Server administration service
- List of JNDI properties for MobileFirst Server live update service
- List of JNDI properties for MobileFirst runtime
- List of JNDI properties for MobileFirst Server push service

Setting up JNDI properties for MobileFirst Server web applications

Set up JNDI properties to configure the MobileFirst Server web applications that are deployed to the application server.
Set the JNDI environment entries in one of the following ways:

- Configure the server environment entries. The steps to configure the server environment entries depends on which application server you use:
    - **WebSphere Application Server:**
        1. In the WebSphere Application Server administration console, go to **Applications → Application Types → WebSphere enterprise applications → application_name → Environment entries for Web modules**.
        2. In the Value fields, enter values that are appropriate to your server environment.



Enterprise Applications

Enterprise Applications > MobileFirst_Runtime > Environment entries for Web modules

Environment entries for Web modules

Configure values for environment entries in web modules.

Web module	URI	Name	Type	Description	Value
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.admin.environmentid	String	[OPTIONAL] Environment identifier. Used to distinguish different instances of Worklight Server that are installed on the same application server or in the same WAS ND cell.	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.admin.jmx.dmgr.host	String	[MANDATORY] [WAS ND only] Deployment Manager host name.	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.admin.jmx.host	String	[MANDATORY] Admin JMX host name.	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	ssl.websphere.alias	String	[OPTIONAL] The WebSphere SSL configuration alias used by the HTTP adapters	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	ssl.websphere.config	String	[OPTIONAL] Set this property to true to have HTTP adapters use WebSphere SSL configuration. Default: false.	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.topology.platform	String	[OPTIONAL] Server type. Possible values: Liberty, WAS, Tomcat. Default: Guessed by the code.	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.authorization.server	String	[MANDATORY] Authorization-server mode - one of the following values: - "embedded" - use the MobileFirst authorization server. - "external" - use an external authorization server. When setting "external" value, you must also set the mfp.external.authorization.server.secret and mfp.external.authorization.server.introspection.url properties for your external server.	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.external.authorization.server.secret	String	[OPTIONAL] Secret of the external authorization server. This property is required when using an external authorization server (mfp.authorization.server = "external"), and is ignored otherwise.	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.external.authorization.server.introspection.url	String	[OPTIONAL] URL of the introspection endpoint of the external authorization server. This property is required when using an external authorization server (mfp.authorization.server = "external"), and is ignored otherwise.	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.admin.jmx.connector	String	[OPTIONAL] [WAS Full Profile only] JMX Connector type. Possible values: SOAP, RMI. Only SOAP is supported in the context of a server farm. Default: SOAP.	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.topology.clustermode	String	[OPTIONAL] Servers topology. Possible values: Standalone, Cluster, Farm. Default: Standalone.	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.admin.jmx.dmgr.port	String	[MANDATORY] [WAS ND only] Deployment Manager SOAP or RMI port (which one, depends on the value of mfp.admin.jmx.connector).	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.admin.jmx.port	String	[MANDATORY] Admin service JMX port	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.admin.jmx.user	String	[MANDATORY] [Liberty Profile] User name for the JMX REST connection. [MANDATORY] [WAS Farm] User name for the SOAP connection.	
mfp-server.war	mfp-server.war,WEB-INF/web.xml	mfp.admin.jmx.pwd	String	[MANDATORY] [Liberty Profile] User password for the JMX REST connection. [MANDATORY] [WAS Farm] User password for the SOAP connection.	

- WebSphere Application Server Liberty:

  In **liberty_install_dir**/**usr**/**servers**/**serverName**, edit the **server.xml** file, and declare the JNDI properties as follows:

```
<application id="app_context_root" name="app_context_root" location="app_war_name.war" type="war">
   ...
</application>
<jndiEntry jndiName="app_context_root/JNDI_property_name" value="JNDI_property_value" />
```

  The context root (in the previous example: **app_context_root**) connects between the JNDI entry and a specific MobileFirst application. If multiple MobileFirst applications exist on the same server, you can define specific JNDI entries for each application by using the context path prefix.

> **Note:** Some properties are defined globally on WebSphere Application Server Liberty, without prefixing the property name by the context root. For a list of these properties, see Global JNDI entries (../appserver/#global-jndi-entries).

  For all other JNDI properties, the names must be prefixed with the context root of the application:

  - For the live update service, the context root must be /**config**. For example, if the context root of the administration service is /**mfpadmin**, then the context root of the live update service must be /**mfpadminconfig**.
  - For the push service, you must define the context root as /**imfpush**. Otherwise, the client devices cannot connect to it as the context root is hardcoded in the SDK.
  - For the MobileFirst Administration Service application, the MobileFirst Operations Console and MobileFirst runtime, you can define the context root as you want. However, by default it is /**mfpadmin** for MobileFirst Administration Service, /**mfpconsole** for MobileFirst Operations Console, and /**mfp** for MobileFirst runtime.

  For example:

```
<application id="mfpadmin" name="mfpadmin" location="mfp-admin-service.war" type="war">
   ...
</application>
<jndiEntry jndiName="mfpadmin/mfp.admin.actions.prepareTimeout" value = "2400000" />
```

- Apache Tomcat:

  1. In **tomcat_install_dir/conf**, edit the **server.xml** file, and declare the JNDI properties as follows:

```
<Context docBase="app_context_root" path="/app_context_root">
   <Environment name="JNDI_property_name" override="false" type="java.lang.String" value="JNDI_property_value"/>
</Context>
```

  - The context path prefix is not needed because the JNDI entries are defined inside the `<Context>` element of an application.
  - `override="false"` is mandatory.
  - The `type` attribute is always `java.lang.String`, unless specified differently for the property.

  For example:

```
<Context docBase="app_context_root" path="/app_context_root">
   <Environment name="mfp.admin.actions.prepareTimeout" override="false" type="java.lang.String" value="2400000"/>
</Context>
```

- If you install with Ant tasks, you can also set the values of the JNDI properties at installation time.

  In **mfp*install*dir/MobileFirstServer/configuration-samples**, edit the configuration XML file for the Ant tasks, and declare the values for the JNDI properties by using the property element inside the following tags:

  - `<installmobilefirstadmin>`, for MobileFirst Server administration, MobileFirst Operations Console, and live update services. For more information, see 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).
  - `<installmobilefirstruntime>`, for MobileFirst runtime configuration properties. For more information, see Ant tasks for installation of MobileFirst runtime environments (../installation-reference/#ant-tasks-for-installation-of-mobilefirst-runtime-environments).
  - `<installmobilefirstpush>`, for configuration of the push service. For more information, see Ant tasks for installation of MobileFirst Server push service (../installation-reference/#ant-tasks-for-installation-of-mobilefirst-server-push-service).

  For example:

```
<installmobilefirstadmin ..>
   <property name = "mfp.admin.actions.prepareTimeout" value = "2400000" />
</installmobilefirstadmin>
```

List of JNDI properties for MobileFirst Server administration service

When you configure MobileFirst Server administration service and MobileFirst Operations Console for your application server, you set optional or mandatory JNDI properties, in particular for Java Management Extensions (JMX).

The following properties can be set on the administration service web application mfp-admin-service.war.

JNDI properties for administration service: JMX

Property	Optional or mandatory	Description	Restrictions
mfp.admin.jmx.connector	Optional	The Java Management Extensions (JMX) connector type. The possible values are `SOAP` and `RMI`. The default value is SOAP.	WebSphere Application Server only.

Property	Optional or mandatory	Description	Restrictions
mfp.admin.jmx.host	Optional	Host name for the JMX REST connection.	Liberty profile only.
mfp.admin.jmx.port	Optional	Port for the JMX REST connection.	Liberty profile only.
mfp.admin.jmx.user	Mandatory for the Liberty profile and for WebSphere Application Server farm, optional otherwise	User name for the JMX REST connection.	WebSphere Application Server Liberty profile: The user name for the JMX REST connection. WebSphere Application Server farm: the user name for the SOAP connection. WebSphere Application Server Network Deployment: the user name of the WebSphere administrator if the virtual host mapped to the MobileFirst server administration application is not the default host. Liberty collective: the user name of the controller administrator that is defined in the `<administrator-role>` element of the server.xml file of the Liberty controller.
mfp.admin.jmx.pwd	Mandatory for the Liberty profile and for WebSphere Application Server farm, optional otherwise	User password for the JMX REST connection.	WebSphere Application Server Liberty profile: the user password for the JMX REST connection. WebSphere Application Server farm: the user password for the SOAP connection. WebSphere Application Server Network Deployment: the user password of the WebSphere administrator if the virtual host that is mapped to the MobileFirst Server server administration application is not the default host. Liberty collective: the password of the controller administrator that is defined in the `<administrator-role>` element of the server.xml file of the Liberty controller.
mfp.admin.rmi.registryPort	Optional	RMI registry port for the JMX connection through a firewall.	Tomcat only.
mfp.admin.rmi.serverPort	Optional	RMI server port for the JMX connection through a firewall.	Tomcat only.
mfp.admin.jmx.dmgr.host	Mandatory	Deployment manager host name.	WebSphere Application Server Network Deployment only.
mfp.admin.jmx.dmgr.port	Mandatory	Deployment manager RMI or SOAP port.	WebSphere Application Server Network Deployment only.

JNDI properties for administration service: timeout

Property	Optional or mandatory	Description
mfp.admin.actions.prepareTimeout	Optional	Timeout in milliseconds to transfer data from the adminstration service to the runtime during a deployment transaction. If the runtime cannot be reached within this time, an error is raised and the deployment transaction ends. Default value: 1800000 ms (30 min)
mfp.admin.actions.commitRejectTimeout	Optional	Timeout in milliseconds, when a runtime is contacted, to commit or reject a deployment transaction. If the runtime cannot be reached within this time, an error is raised and the deployment transaction ends. Default value: 120000 ms (2 min)
mfp.admin.lockTimeoutInMillis	Optional	Timeout in milliseconds for obtaining the transaction lock. Because deployment transactions run sequentially, they use a lock. Therefore, a transaction must wait until a previous transaction is finished. This timeout is the maximal time during which a transaction waits. Default value: 1200000 ms (20 min)
mfp.admin.maxLockTimeInMillis	Optional	The maximal time during which a process can take the transaction lock. Because deployment transactions run sequentially, they use a lock. If the application server fails while a lock is taken, it can happen in rare situations that the lock is not released at the next restart of the application server. In this case, the lock is released automatically after the maximum lock time so that the server is not blocked forever. Set a time that is longer than a normal transaction. Default value: 1800000 (30 min)

JNDI properties for administration service: logging

Property	Optional or mandatory	Description
mfp.admin.logging.formatjson	Optional	Set this property to true to enable pretty formatting (extra blank space) of JSON objects in responses and log messages. Setting this property is helpful when you debug the server. Default value: false.
mfp.admin.logging.tosystemerror	Optional	Specifies whether all logging messages are also directed to System.Error. Setting this property is helpful when you debug the server.

JNDI properties for administration service: proxies

Property	Optional or mandatory	Description
mfp.admin.proxy.port	Optional	If the MobileFirst administration server is behind a firewall or reverse proxy, this property specifies the address of the host. Set this property to enable a user outside the firewall to reach the MobileFirst administration server. Typically, this property is the port of the proxy, for example 443. It is necessary only if the protocol of the external and internal URIs are different.
mfp.admin.proxy.protocol	Optional	If the MobileFirst administration server is behind a firewall or reverse proxy, this property specifies the protocol (HTTP or HTTPS). Set this property to enable a user outside the firewall to reach the MobileFirst administration server. Typically, this property is set to the protocol of the proxy. For example, wl.net. This property is necessary only if the protocol of the external and internal URIs are different.
mfp.admin.proxy.scheme	Optional	This property is just an alternative name for mfp.admin.proxy.protocol.

Property	Optional or mandatory	Description
mfp.admin.proxy.host	Optional	If the MobileFirst administration server is behind a firewall or reverse proxy, this property specifies the address of the host. Set this property to enable a user outside the firewall to reach the MobileFirst administration server. Typically, this property is the address of the proxy.

JNDI properties for administration service: topologies

Property	Optional or mandatory	Description
mfp.admin.audit	Optional.	Set this property to false to disable the audit feature of the MobileFirst Operations Console. The default value is true.
mfp.admin.environmentid	Optional.	The environment identifier for the registration of the MBeans. Use this identifier when different instances of the MobileFirst Server are installed on the same application server. The identifier determines which administration service, which console, and which runtimes belong to the same installation. The administration service manages only the runtimes that have the same environment identifier.
mfp.admin.serverid	Mandatory for server farms and Liberty collective, optional otherwise.	Server farm: the server identifier. Must be different for each server in the farm. Liberty collective: the value must be controller.
mfp.admin.hsts	Optional.	Set to true to enable HTTP Strict Transport Security according to RFC 6797.
mfp.topology.platform	Optional	Server type. Valid values: • Liberty • WAS • Tomcat If you do not set the value, the application tries to guess the server type.
mfp.topology.clustermode	Optional	In addition to the server type, specify here the server topology. Valid values: • Standalone • Cluster • Farm The default value is Standalone.
mfp.admin.farm.heartbeat	Optional	This property enables you to set in minutes the heartbeat rate that is used in server farm topologies. The default value is 2 minutes. In a server farm, all members must use the same heartbeat rate. If you set or change this JNDI value on one server in the farm, you must also set the same value on every other server in the farm. For more information, see Lifecycle of a server farm node (../appserver/#lifecycle-of-a-server-farm-node).
mfp.admin.farm.missed.heartbeats.timeout	Optional	This property enables you to set the number of missed heartbeats of a farm member before the status of the farm member is considered to be failed or down. The default value is 2. In a server farm all members must use the same missed heartbeat value. If you set or change this JNDI value on one server in the farm, you must also set the same value on every other server in the farm. For more information, see Lifecycle of a server farm node (../appserver/#lifecycle-of-a-server-farm-node).

Property	Optional or mandatory	Description
mfp.admin.farm.reinitialize	Optional	A Boolean value (true or false) for re-registering or re-initializing the farm member.
mfp.swagger.ui.url	Optional	This property defines the URL of the Swagger user interface to be displayed in the administration console.

JNDI properties for administration service: relational database

Property	Optional or mandatory	Description
mfp.admin.db.jndi.name	Optional	The JNDI name of the database. This parameter is the normal mechanism to specify the database. The default value is **java:comp/env/jdbc/mfpAdminDS**.
mfp.admin.db.openjpa.ConnectionDriverName	Optional/Conditionally mandatory	The fully qualified name of the database connection driver class. Mandatory only when the data source that is specified by the **mfp.admin.db.jndi.name** property is not defined in the application server configuration.
mfp.admin.db.openjpa.ConnectionURL	Optional/Conditionally mandatory	The URL for the database connection. Mandatory only when the data source that is specified by the **mfp.admin.db.jndi.name** property is not defined in the application server configuration.
mfp.admin.db.openjpa.ConnectionUserName	Optional/Conditionally mandatory	The user name for the database connection. Mandatory only when the data source that is specified by the **mfp.admin.db.jndi.name** property is not defined in the application server configuration.
mfp.admin.db.openjpa.ConnectionPassword	Optional/Conditionally mandatory	The password for the database connection. Mandatory only when the data source that is specified by the **mfp.admin.db.jndi.name** property is not defined in the application server configuration.
mfp.admin.db.openjpa.Log	Optional	This property is passed to OpenJPA and enables JPA logging. For more information, see the Apache OpenJPA User's Guide (http://openjpa.apache.org/docs/openjpa-0.9.0-incubating/manual/manual.html).
mfp.admin.db.type	Optional	This property defines the type of database. The default value is inferred from the connection URL.

JNDI properties for administration service: licensing

Property	Optional or mandatory	Description
mfp.admin.license.key.server.host	• Optional for perpetual licenses • Mandatory for token licenses	Host name of the Rational License Key Server.
mfp.admin.license.key.server.port	• Optional for perpetual licenses • Mandatory for token licenses	Port number of the Rational License Key Server.

JNDI properties for administration service: JNDI configurations

Property	Optional or mandatory	Description
mfp.jndi.configuration	Optional	The name of the JNDI configuration if the JNDI properties (except this one) must be read from a property file that is injected into the WAR file. If you do not set this property, JNDI properties are not read from a property file.

Property	Optional or mandatory	Description
mfp.jndi.file	Optional	The name of the file that contains the JNDI configuration if the JNDI properties (except this one) must be read from a file installed in the web server. If you do not set this property, JNDI properties are not read from a property file.

The administration service uses a live update service as an auxiliary facility to store various configurations. Use these properties to configure how to reach the live update service.

JNDI properties for administration service: live update service

Property	Optional or mandatory	Description
mfp.config.service.url	Optional The URL of the live update service. The default URL is derived from the URL of administration service by adding config to the context root of the administration service.	
mfp.config.service.user	Mandatory	The user name that is used to access the live update service. In a server farm topology, the user name must be the same for all the members of the farm.
mfp.config.service.password	Mandatory	The password that is used to access the live update service. In a server farm topology, the password must be the same for all the members of the farm.
mfp.config.service.schema	Optional	The name of the schema that is used by the live update service.

The administration service uses a push service as an auxiliary facility to store various push settings. Use these properties to configure how to reach the push service. Because the push service is protected by the OAuth security model, you must set various properties to enable confidential clients in OAuth.

JNDI properties for administration service: push service

Property	Optional or mandatory	Description
mfp.admin.push.url	Optional	The URL of the push service. If the property is not specified, the push service is considered disabled. If the property is not properly set, the administration service cannot contact the push service and the administration of push services in MobileFirst Operations Console does not work.
mfp.admin.authorization.server.url	Optional	The URL of the OAuth authorization server that is used by the push service. The default URL is derived from the URL of the administration service by changing the context root to the context root of the first installed runtime. If you install multiple runtimes, it is best to set the property. If the property is not set properly, the administration service cannot contact the push service and the administration of push services in MobileFirst Operations Console does not work.
mfp.push.authorization.client.id	Optional/conditionally mandatory	The identifier of the confidential client that handles OAuth authorization for the push service. Mandatory only if the **mfp.admin.push.url** property is specified.
mfp.push.authorization.client.secret	Optional/conditionally mandatory	The secret of the confidential client that handles OAuth authorization for the push service. Mandatory only if the **mfp.admin.push.url** property is specified
mfp.admin.authorization.client.id	Optional/conditionally mandatory	The identifier of the confidential client that handles OAuth authorization for the administration service. Mandatory only if the **mfp.admin.push.url** property is specified.

Property	Optional or mandatory	Description
mfp.push.authorization.client.secret	Optional/conditionally mandatory	The secret of the confidential client that handles OAuth authorization for the administration service. Mandatory only if the **mfp.admin.push.url** property is specified.

JNDI properties for MobileFirst Operations Console

The following properties can be set on the web application (mfp-admin-ui.war) of MobileFirst Operations Console.

Property	Optional or mandatory	Description
mfp.admin.endpoint	Optional	Enables the MobileFirst Operations Console to locate the MobileFirst Server administration REST service. Specify the external address and context root of the **mfp-admin-service.war** web application. In a scenario with a firewall or a secured reverse proxy, this URI must be the external URI and not the internal URI inside the local LAN. For example, https://wl.net:443/mfpadmin.
mfp.admin.global.logout	Optional	Clears the WebSphere user authentication cache during the console logout. This property is useful only for WebSphere Application Server V7. The default value is false.
mfp.admin.hsts	Optional	Set this property to true to enable HTTP Strict Transport Security (http://www.w3.org/Security/wiki/Strict_Transport_Security) according to RFC 6797. For more information, see the W3C Strict Transport Security page. The default value is false.
mfp.admin.ui.cors	Optional	The default value is true. For more information, see the W3C Cross-Origin Resource Sharing page (http://www.w3.org/TR/cors/).
mfp.admin.ui.cors.strictssl	Optional	Set to false to allow CORS situations where the MobileFirst Operations Console is secured with SSL (HTTPS protocol) while the MobileFirst Server administration service is not, or conversely. This property takes effect only if the **mfp.admin.ui.cors** property is enabled.

List of JNDI properties for MobileFirst Server live update service

When you configure the MobileFirst Server live update service for your application server, you can set the following JNDI properties. The table lists the JNDI properties for the IBM relational database live update service.

Property	Optional or mandatory	Description
mfp.db.relational.queryTimeout	Optional	Timeout for executing a query in RDBMS, in seconds. A value of zero means an infinite timeout. A negative value means the default (no override). In case no value is configured, a default value is used. For more information, see setQueryTimeout (http://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html#setQueryTimeout(int)).

To know how to set those properties, see Setting up JNDI properties for MobileFirst Server web applications.

List of JNDI properties for MobileFirst runtime

When you configure the MobileFirst Server runtime for your application server, you need to set the optional or mandatory JNDI properties. The following table lists the MobileFirst properties that are always available as JNDI entries:

Property	Description
mfp.admin.jmx.dmgr.host	Mandatory. The host name of the deployment manager. WebSphere Application Server Network Deployment only.
mfp.admin.jmx.dmgr.port	Mandatory. The RMI or SOAP port of the deployment manager. WebSphere Application Server Network Deployment only.

Property	Description
mfp.admin.jmx.host	Liberty only. The host name for the JMX REST connection. For Liberty collective, use the host name of the controller.
mfp.admin.jmx.port	Liberty only. The port number for the JMX REST connection. For Liberty collective, the port of the REST connector must be identical to the value of the httpsPort attribute that is declared in the `<httpEndpoint>` element. This element is declared in the server.xml file of the Liberty controller.
mfp.admin.jmx.user	Optional. WebSphere Application Server farm: the user name of the SOAP connection. Liberty collective: the user name of the controller administrator that is defined in the `<administrator-role>` element of the server.xml file of the Liberty controller.
mfp.admin.jmx.pwd	Optional. WebSphere Application Server farm: the user passsword of the SOAP connection. Liberty collective: the password of the controller administrator that is defined in the `<administrator-role>` element of the server.xml file of the Liberty controller.
mfp.admin.serverid	Mandatory for server farms and Liberty collective, optional otherwise. Server farm: the server identifier. Must be different for each server in the farm. Liberty collective: the member identifier. The identifier must be different for each member in the collective. The value controller cannot be used as it is reserved for the collective controller.
mfp.topology.platform	Optional. The server type. Valid values are: • Liberty • WAS • Tomcat If you do not set the value, the application tries to guess the server type.
mfp.topology.clustermode	Optional. In addition to the server type, specify here the server topology. Valid values: • Standalone • Cluster • Farm The default value is Standalone.
mfp.admin.jmx.replica	Optional. For Liberty collective only. Set this property only when the administration components that manage this runtime are deployed in different Liberty controllers (replicas). Endpoint list of the different controller replicas with the following syntax: `replica-1 hostname:replica-1 port, replica-2 hostname:replica-2 port,..., replica-n hostname:replica-n port`
mfp.analytics.console.url	Optional. The URL that is exposed by IBM MobileFirst Analytics that links to the Analytics console. Set this property if you want to access the Analytics console from the MobileFirst Operations Console. For example, `http://<hostname>:<port>/analytics/console`
mfp.analytics.password	The password that is used if the data entry point for the IBM MobileFirst Analytics is protected with basic authentication.
mfp.analytics.url	The URL that is exposed by the IBM MobileFirst Analytics that receives incoming analytics data. For example, `http://<hostname>:<port>/analytics-service/rest`
mfp.analytics.username	The user name that is used if the data entry point for the IBM MobileFirst Analytics is protected with basic authentication.

Property	Description
mfp.device.decommissionProcessingInterval	Defines how often (in seconds) the decommissioning task is executed. Default: 86400, which is one day.
mfp.device.decommission.when	The number of days of inactivity after which a client device is decommissioned by the device decommissioning task. Default: 90 days.
mfp.device.archiveDecommissioned.when	The number of days of inactivity, after which a client device that has been decommissioned is archived. This task writes the client devices that were decommissioned to an archive file. The archived client devices are written to a file in the MobileFirst Server **home\devices_archive** directory. The name of the file contains the time stamp when the archive file is created. Default: 90 days.
mfp.licenseTracking.enabled	A value that is used to enable or disable device tracking in IBM MobileFirst Foundation. For performance reasons, you can disable device tracking when IBM MobileFirst Foundation runs only Business-to-Consumer (B2C) apps. When device tracking is disabled, the license reports are also disabled and no license metrics are generated. Possible values are true (default) and false.
mfp.runtime.temp.folder	Defines the runtime temporary files folder. Uses the default temporary folder location of the web container when not set.
mfp.adapter.invocation.url	The URL to be used for invoking adapter procedures from inside Java adapters, or JavaScript adapters that are invoked using the rest endpoint. If this property is not set, the URL of the currently executing request will be used (this is the default behavior). This value should contain the full URL, including the context root.
mfp.authorization.server	Authorization-server mode. Can be one of the following mode: • embedded: Use the MobileFirst authorization server. • external: Use an external authorization server . When setting this value, you must also set the **mfp.external.authorization.server.secret** and **mfp.external.authorization.server.introspection.url** properties for your external server.
mfp.external.authorization.server.secret	Secret of the external authorization server. This property is required when using an external authorization server, meaning **mfp.authorization.server** is set to external and is ignored otherwise.
mfp.external.authorization.server.introspection.url	URL of the introspection endpoint of the external authorization server. This property is required when using an external authorization server, meaning **mfp.authorization.server** is set to **external** and is ignored otherwise.
ssl.websphere.config	Used to configure the keystore for an HTTP adapter. When set to false (default), instructs the MobileFirst runtime to use the MobileFirst keystore. When set to true, instructs the MobileFirst runtime to use the WebSphere SSL configuration. For more information, see WebSphere Application Server SSL configuration and HTTP adapters.

List of JNDI properties for MobileFirst Server push service

Property	Optional or mandatory	Description
mfp.push.db.type	Optional	Database type. Possible values: DB, CLOUDANT. Default: DB
mfp.push.db.queue.connections	Optional	Number of threads in the thread pool that does the database operation. Default: 3

Property	Optional or mandatory	Description
mfp.push.db.cloudant.url	Optional	The Cloudant account URL. When this property is defined, the Cloudant DB will be directed to this URL.
mfp.push.db.cloudant.dbName	Optional	The name of the database in the Cloudant account. It must start with a lowercase letter and consist only of lowercase letters, digits, and the characters _, $, and -. Default: mfp_push_db
mfp.push.db.cloudant.username	Optional	The user name of the Cloudant account, used to store the database. when this property is not defined, a relational database is used.
mfp.push.db.cloudant.password	Optional	The password of the Cloudant account, used to store the database. This property must be set when mfp.db.cloudant.username is set.
mfp.push.db.cloudant.doc.version	Optional	The Cloudant document version.
mfp.push.db.cloudant.socketTimeout	Optional	A timeout for detecting the loss of a network connection for Cloudant, in milliseconds. A value of zero means an infinite timeout. A negative value means the default (no override). Default. See https://github.com/cloudant/java-cloudant#advanced-configuration (https://github.com/cloudant/java-cloudant#advanced-configuration).
mfp.push.db.cloudant.connectionTimeout	Optional	A timeout for establishing a network connection for Cloudant, in milliseconds. A value of zero means an infinite timeout. A negative value means the default (no override). Default. See https://github.com/cloudant/java-cloudant#advanced-configuration (https://github.com/cloudant/java-cloudant#advanced-configuration).
mfp.push.db.cloudant.maxConnections	Optional	The Cloudant connector's max connections. Default. See https://github.com/cloudant/java-cloudant#advanced-configuration (https://github.com/cloudant/java-cloudant#advanced-configuration).
mfp.push.db.cloudant.ssl.authentication	Optional	A Boolean value (true or false) that specifies whether the SSL certificate chain validation and host name verification are enabled for HTTPS connections to the Cloudant database. Default: True
mfp.push.db.cloudant.ssl.configuration	Optional	(WAS Full Profile only) For HTTPS connections to the Cloudant database: The name of an SSL configuration in the WebSphere Application Server configuration, to use when no configuration is specified for the host and port.
mfp.push.db.cloudant.proxyHost	Optional	Cloudant connector's proxy host. Default: See https://github.com/cloudant/java-cloudant#advanced-configuration (https://github.com/cloudant/java-cloudant#advanced-configuration).
mfp.push.db.cloudant.proxyPort	Optional	Cloudant connector's proxy port. Default: See https://github.com/cloudant/java-cloudant#advanced-configuration (https://github.com/cloudant/java-cloudant#advanced-configuration).
mfp.push.services.ext.security	Optional	The security extension plugin.
mfp.push.security.endpoint	Optional	The endpoint URL for the authorization server.
mfp.push.security.user	Optional	The username to access the authorization server.
mfp.push.security.password	Optional	The password to access the authorization server.
mfp.push.services.ext.analytics	Optional	The analytics extension plugin.
mfp.push.analytics.endpoint	Optional	The endpoint URL for the analytics server.
mfp.push.analytics.user	Optional	The username to access the analytics server.
mfp.push.analytics.password	Optional	The password to access the analytics server.
mfp.push.analytics.events.appCreate	Optional	The analytic event when the application is created. Default: true

Property	Optional or mandatory	Description
mfp.push.analytics.events.appDelete	Optional	The analytic event when the application is deleted. Default: true
mfp.push.analytics.events.deviceRegister	Optional	The analytic event when the device is registered. Default: true
mfp.push.analytics.events.deviceUnregister	Optional	The analytic event when the device is unregistered. Default: true
mfp.push.analytics.events.tagSubscribe	Optional	The analytic event when the device is subscribed to tag. Default: true
mfp.push.analytics.events.tagUnsubscribe	Optional	The analytic event when the device is unsubscribed from tag. Default: true
mfp.push.analytics.events.notificationSendSuccess	Optional	The analytic event when the notification is sent successfully. Default: true
mfp.push.analytics.events.notificationSendFailure	Optional	The analytic event when the notification is failed to send. Default: false
mfp.push.analytics.events.inactiveDevicePurge	Optional	The analytic event when the inactive devices are deleted. Default: true
mfp.push.analytics.events.msgReqAccepted	Optional	The analytic event when the notification is accepted for delivery. Default: true
mfp.push.analytics.events.msgDispatchFailed	Optional	The analytic event when the notification dispatch failed. Default: true
mfp.push.analytics.events.notificationDispatch	Optional	The analytic event when the notification is about to be dispatched. Default: true
mfp.push.internalQueue.maxLength	Optional	The length of the queue which holds the notification tasks before dispatch. Default: 200000
mfp.push.gcm.proxy.enabled	Optional	Shows whether Google GCM must be accessed through a proxy. Default: false
mfp.push.gcm.proxy.protocol	Optional	Can be either http or https.
mfp.push.gcm.proxy.host	Optional	GCM proxy host. Negative value means default port.
mfp.push.gcm.proxy.port	Optional	GCM proxy port. Default: -1
mfp.push.gcm.proxy.user	Optional	Proxy user name, if the proxy requires authentication. Empty user name means no authentication.
mfp.push.gcm.proxy.password	Optional	Proxy password, if the proxy requires authentication.
mfp.push.gcm.connections	Optional	Push GCM max connections. Default : 10
mfp.push.apns.proxy.enabled	Optional	Shows whether APNs must be accessed through a proxy. Default: false
mfp.push.apns.proxy.type	Optional	APNs proxy type.
mfp.push.apns.proxy.host	Optional	APNs proxy host.
mfp.push.apns.proxy.port	Optional	APNs proxy port. Default: -1
mfp.push.apns.proxy.user	Optional	Proxy user name, if the proxy requires authentication. Empty user name means no authentication.
mfp.push.apns.proxy.password	Optional	Proxy password, if the proxy requires authentication.
mfp.push.apns.connections	Optional	Push APNs max connections. Default : 3
mfp.push.apns.connectionIdleTimeout	Optional	APNs Idle Connection Timeout. Default : 0

Configuring data sources

Find out some data source configuration details pertaining to the supported databases.

- Managing the DB2 transaction log size
- Configuring DB2 HADR seamless failover for MobileFirst Server and Application Center data sources
- Handling stale connections
- Stale data after creating or deleting apps from MobileFirst Operations Console

Managing the DB2 transaction log size

When you deploy an application that is at least 40 MB with IBM MobileFirst Operations Console, you might receive a transaction log full error.

The following system output is an example of the transaction log full error code.

```
DB2 SQL Error: SQLCODE=-964, SQLSTATE=57011
```

The content of each application is stored in the MobileFirst administration database.

The active log files are defined in number by the **LOGPRIMARY** and **LOGSECOND** database configuration parameters, and in size by the **LOGFILSIZ** database configuration parameter. A single transaction cannot use more log space than **LOGFILSZ** * (**LOGPRIMARY** + **LOGSECOND**) * 4096 KB.

The `DB2 GET DATABASE CONFIGURATION` command includes information about the log file size, and the number of primary and secondary log files.

Depending on the largest size of the MobileFirst application that is deployed, you might need to increase the DB2 log space.

Using the `DB2 update db cfg` command, increase the **LOGSECOND** parameter. Space is not allocated when the database is activated. Instead, the space is allocated only as needed.

Configuring DB2 HADR seamless failover for MobileFirst Server and Application Center data sources

You must enable the seamless failover feature with WebSphere Application Server Liberty profile and WebSphere Application Server. With this feature, you can manage an exception when a database fails over and gets rerouted by the DB2 JDBC driver.

> **Note:** DB2 HADR failover is not supported for Apache Tomcat.

By default with DB2 HADR, when the DB2 JDBC driver performs a client reroute after detecting that a database failed over during the first attempt to reuse an existing connection, the driver triggers **com.ibm.db2.jcc.am.ClientRerouteException**, with **ERRORCODE=-4498** and **SQLSTATE=08506**. WebSphere Application Server maps this exception to **com.ibm.websphere.ce.cm.StaleConnectionException** before it is received by the application.

In this case, the application would have to catch the exception and execute again the transaction. The MobileFirst and Application Center runtime environments do not manage the exception but rely on a feature that is called seamless failover. To enable this feature, you must set the **enableSeamlessFailover** JDBC property to "1".

WebSphere Application Server Liberty profile configuration

You must edit the **server.xml** file, and add the **enableSeamlessFailover** property to the **properties.db2.jcc** element of the MobileFirst and Application Center data sources. For example:

```
<dataSource jndiName="jdbc/WorklightAdminDS" transactional="false">
  <jdbcDriver libraryRef="DB2Lib"/>
  <properties.db2.jcc databaseName="WLADMIN"  currentSchema="WLADMSC"
                      serverName="db2server" portNumber="50000"
                      enableSeamlessFailover= "1"
                      user="worklight" password="worklight"/>
</dataSource>
```

WebSphere Application Server configuration

From the WebSphere Application Server administrative console for each MobileFirst and Application Center data source:

1. Go to **Resources → JDBC → Data sources → DataSource name** .
2. Select **New** and add the following custom property, or update the values if the properties already exist: `enableSeamlessFailover : 1`
3. Click **Apply**.
4. Save your configuration.

For more information about how to configure a connection to an HADR-enabled DB2 database, see Setting up a connection to an HADR-enabled DB2 database (https://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/tdat_db2_hadr.html?cp=SSAW57_8.5.5%2F3-3-6-3-3-0-7-3&lang=en).

Handling stale connections

Configure your application server to avoid database timeout issues.

A **StaleConnectionException** is an exception that is generated by the Java application server profile database connection code when a JDBC driver returns an unrecoverable error from a connection request or operation. The **StaleConnectionException** is raised when the database vendor issues an exception to indicate that a connection currently in the connection pool is no longer valid. This exception can happen for many reasons. The most common cause of **StaleConnectionException** is due to retrieving connections from the database connection pool and finding out that the connection has timed out or dropped when it was unused for a long time.

You can configure your application server to avoid this exception.

Apache Tomcat configuration

MySQL
The MySQL database closes its connections after a period of non-activity on a connection. This timeout is defined by the system variable called **wait_timeout**. The default is 28000 seconds (8 hours).

When an application tries to connect to the database after MySQL closes the connection, the following exception is generated:

```
com.mysql.jdbc.exceptions.jdbc4.MySQLNonTransientConnectionException: No operations allowed after statement closed.
```

Edit the **server.xml** and **context.xml** files, and for every `<Resource>` element add the following properties:

- **testOnBorrow="true"**
- **validationQuery="select 1"**

For example:

```
<Resource name="jdbc/AppCenterDS"
  type="javax.sql.DataSource"
  driverClassName="com.mysql.jdbc.Driver"
  ...
  testOnBorrow="true"
  validationQuery="select 1"
/>
```

WebSphere Application Server Liberty profile configuration

Edit the **server.xml** file and for every `<dataSource>` element (runtime and Application Center databases) add a `<connectionManager>` element with the agedTimeout property:

```
<connectionManager agedTimeout="timeout_value"/>
```

The timeout value depends mainly on the number of opened connections in parallel but also on the minimum and maximum number of the connections in the pool. Hence, you must tune the different **connectionManager** attributes to identify the most adequate values. For more information about the **connectionManager** element, see Liberty: Configuration elements in the **server.xml** file (https://www.ibm.com/support/knowledgecenter/SSD28V_8.5.5/com.ibm.websphere.wlp.core.doc/autodita/rwlp_metatype_core.html).

> **Note:** MySQL in combination with WebSphere Application Server Liberty profile or WebSphere Application Server full profile is not classified as a supported configuration. For more information, see WebSphere Application Server Support Statement (http://www.ibm.com/support/docview.wss?uid=swg27004311). Use IBM DB2 or another database that is supported by WebSphere Application Server to benefit from a configuration that is fully supported by IBM Support.

Stale data after creating or deleting apps from MobileFirst Operations Console

On a Tomcat 8 application server, if you use a MySQL database, some calls from MobileFirst Operations Console to services return a 404 error.

On a Tomcat 8 application server, if you work with a MySQL database, when you use MobileFirst Operations Console to delete an app, or add a new one, and try to refresh the console a couple of times, you might see stale data. For example, users might see an already deleted app in the list.

To avoid this problem, change the isolation level to **READ_COMMITTED**, either in the data source, or in the database management system.

For the meaning of **READ_COMMITTED**, see the MySQL documentation (http://www.ibm.com/doc/refman/5.7/en/innodb-transaction-isolation-levels.html?view=kc) at http://dev.mysql.com/doc/refman/5.7/en/innodb-transaction-isolation-levels.html (http://dev.mysql.com/doc/refman/5.7/en/innodb-transaction-isolation-levels.html).

- To change the isolation level to **READ_COMMITTED** in the data source, modify the **server.xml** Tomcat configuration file: In the section, add the **defaultTransactionIsolation="READ_COMMITTED"** attribute.
- To change the isolation level to **READ_COMMITTED** globally in the database management system, refer to the SET TRANSACTION Syntax page (http://dev.mysql.com/doc/refman/5.7/en/set-transaction.html) of the MySQL documentation at http://dev.mysql.com/doc/refman/5.7/en/set-transaction.html (http://dev.mysql.com/doc/refman/5.7/en/set-transaction.html).

WebSphere Application Server full profile configuration

DB2 or Oracle
To minimize the stale connection issues, check the connection pools configuration on each data source in WebSphere Application Server administration console.

1. Log in to the WebSphere Application Server administration console.
2. Select **Resources → JDBC Providers → database *jdbc*provider → Data Sources → your*data*source → Connection pool properties**.
3. Set the **Minimum connections** value to 0.
4. Set the **Reap time** value to be lesser than the **Unused timeout** value.
5. Make sure that the **Purge policy** property is set to **EntirePool (default)**.

For more information, see Connection pool settings (https://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/udat_conpoolset.html).

MySQL

1. Log in to the WebSphere Application Server administration console.
2. Select **Resources → JDBC → Data sources**.
3. For each MySQL data source:
    - Click the data source.
    - Select **Connection pool** properties under **Additional Properties**.
    - Modify the value of the **Aged timeout** property. The value must be lower than the MySQL **wait_timeout** system variable so that the connections are purged before MySQL closes these connections.
    - Click **OK**.

> **Note:** MySQL in combination with WebSphere Application Server Liberty profile or WebSphere Application Server full profile is not classified as a supported configuration. For more information, see WebSphere Application Server Support Statement (http://www.ibm.com/support/docview.wss?uid=swg27004311). Use IBM DB2 or another database that is supported by WebSphere Application Server to benefit from a configuration that is fully supported by IBM Support.

Configuring logging and monitoring mechanisms

IBM MobileFirst Foundation reports errors, warnings, and informational messages into a log file. The underlying logging mechanism varies by application server.

MobileFirst Server

IBM MobileFirst Platform Server (MobileFirst Server for short) uses the standard java.util.logging package. By default, all MobileFirst logging goes to the application server log files. You can control MobileFirst Server logging by using the standard tools that are available in each application server. For example, if you want to activate trace logging in WebSphere Application Server Liberty, add a trace element to the server.xml file. To activate trace logging in WebSphere Application Server, use the logging screen in the console and enable trace for MobileFirst logs.

MobileFirst logs all begin with **com.ibm.mfp**.
Application Center logs begin with **com.ibm.puremeap**.

For more information about the logging models of each application server, including the location of the log files, see the documentation for the relevant application server, as shown in the following table.

Application server	Location of documentation
Apache Tomcat	http://tomcat.apache.org/tomcat-7.0-doc/logging.html#Using*java.util.logging*(default) (http://tomcat.apache.org/tomcat-7.0-doc/logging.html#Using_java.util.logging_(default))

Application server	Location of documentation
WebSphere Application Server Version 8.5 full profile	http://ibm.biz/knowctr#SSEQTP*8.5.5/com.ibm.websphere.base.doc/ae/ttrb*trcover.html (http://ibm.biz/knowctr#SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/ttrb_trcover.html)
WebSphere Application Server Version 8.5 Liberty profile	http://ibm.biz/knowctr#SSEQTP*8.5.5/com.ibm.websphere.wlp.doc/ae/rwlp*logging.html?cp=SSEQTP_8.5.5%2F1-16-0-0 (http://ibm.biz/knowctr#SSEQTP_8.5.5/com.ibm.websphere.wlp.doc/ae/rwlp_logging.html?cp=SSEQTP_8.5.5%2F1-16-0-0)

Log level mappings

MobileFirst Server uses the **java.util.logging** API. The logging levels map to the following levels:

- WL.Logger.debug: FINE
- WL.Logger.info: INFO
- WL.Logger.warn: WARNING
- WL.Logger.error: SEVERE

Log monitoring tools

For Apache Tomcat, you can use IBM Operations Analytics - Log Analysis (http://www.ibm.com/software/products/en/ibm-operations-analytics---log-analysis) or other industry standard log file monitoring tools to monitor logs and highlight errors and warnings.

For WebSphere Application Server, use the log viewing facilities that are described in IBM Knowledge Center. The URLs are listed in the table in the MobileFirst Server section of this page.

Back-end connectivity

To enable trace to monitor back-end connectivity, see the documentation for your specific application server platform in the table of section MobileFirst Server of this page. Use the **com.ibm.mfp.server.js.adapter** package and set the log level to **FINEST**.

Audit log for administration operations

MobileFirst Operations Console stores an audit log for login, logout, and for all administration operations, such as deploying apps or adapters or locking apps. You can disable the audit log by setting the JNDI property **mfp.admin.audit** to false on the web application of the MobileFirst administration service (**mfp-admin-service.war**).

When the audit log is enabled, you can download it from MobileFirst Operations Console by clicking the **Audit log** link in the footer of the page.

Login and authentication issues

To diagnose login and authentication issues, enable the package **com.ibm.mfp.server.security** for trace and set the log level to **FINEST**.

Configuring multiple runtimes

You can configure MobileFirst Server with multiple runtimes, creating a visual differentiation between application "types" in the MobileFirst Operations Console.

> **Note:** multiple runtimes are not supported in a Mobile Foundation server instance created by the Mobile Foundation Bluemix service. In the Bluemix service, you must create multiple service instances instead.

Jump to

Configuring multiple runtimes in WebSphere Liberty profile

1. Open the **server.xml** file of the application server. Typically located in the **[application-server]/usr/servers/server-name/** folder. For example, with the MobileFirst Developer Kit, the file can be found it **[installation-folder]/mfp-server/usrs/servers/mfp/server.xml**.
2. Add a second `application` element:

```
<application id="second-runtime" name="second-runtime" location="mfp-server.war" type="war">
  <classloader delegation="parentLast">
    </classloader>
</application>
```

3. Add a second set of JNDI entries:

```
<jndiEntry jndiName="second-runtime/mfp.analytics.url" value="'http://localhost:9080/analytics-service/rest'"/>
<jndiEntry jndiName="second-runtime/mfp.analytics.console.url" value="'http://localhost:9080/analytics/console'"/>
<jndiEntry jndiName="second-runtime/mfp.analytics.username" value="'admin'"/>
<jndiEntry jndiName="second-runtime/mfp.analytics.password" value="'admin'"/>
<jndiEntry jndiName="second-runtime/mfp.authorization.server" value="'embedded'"/>
```

4. Add a second `dataSource` element:

```
<dataSource jndiName="second-runtime/jdbc/mfpDS" transactional="false">
  <jdbcDriver libraryRef="DerbyLib"/>
  <properties.derby.embedded databaseName="${wlp.install.dir}/databases/second-runtime" user="'MFPDATA'"/>
</dataSource>
```

> **Note:**
>
> - Make sure the `dataSource` is pointing to a different database schema.
> - Make sure you have created another database instance (../databases) for the new runtime.
> - In the development environment, add `createDatabase="create"` in the `properties.derby.embedded` child-element.

5. Restart the application server.

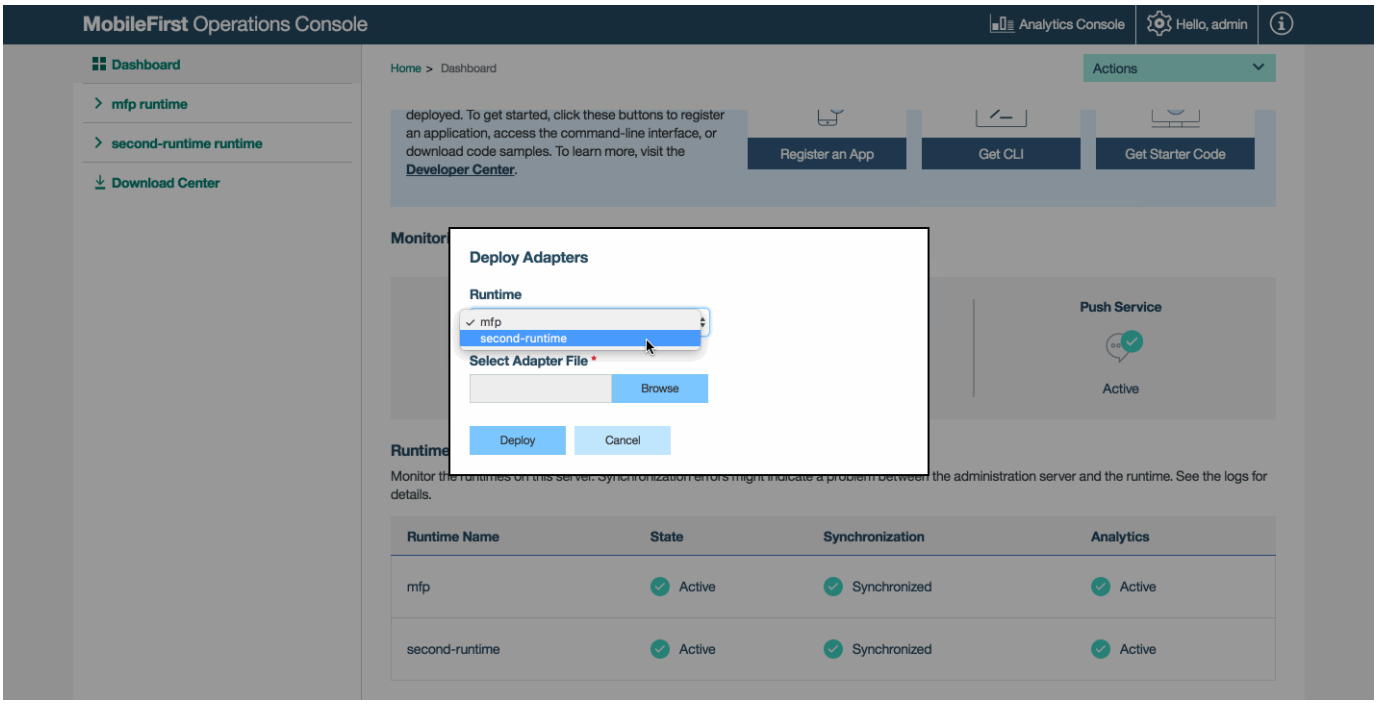
Registering applications and deploying adapters to different runtimes

When a MobileFirst Server is configured with multiple runtimes, the registration of applications and deployment of adapters is slightly different.

- Registering and deploying from the MobileFirst Operations Console
- Registering and deploying from the Command-line

Registering and deploying from the MobileFirst Operations Console

When performing these actions in the MobileFirst Console, you now need to select the runtime to register or deploy to.



Registering and deploying from the Command-line

When performing these actions using the **mfpdev** command-line tool, you now need to add the runtime name to register or deploy to.

To register an application: `mfpdev app register <server-name> <runtime-name>`.

```
mfpdev app register local second-runtime
```

To deploy an adapter: `mfpdev adapter deploy <server-name> <runtime-name>`.

```
mfpdev adapter deploy local second-runtime
```

- **local** is the name of the default server definition in the MobileFirst CLI. Replace *local* with a the server definition name you need to register or deploy to.
- **runtime-name** is the name of the runtime to register or deploy to.

> Learn more with the following CLI help commands:
>
> - `mfpdev help server add`
> - `mfpdev help app register`
> - `mfpdev help adapter deploy`

Exporting and importing runtime configurations

You can export a runtime configuration and import it to another MobileFirst Server using the REST APIs of the MobileFirst Server **administration service**.

For example, you can setup a runtime configuration in a development environment, export its configuration and then import it to a testing environment for a quick set-up, and then further configure it for the specific needs of the testing environment.

> Find out all available REST APIs in the API Reference (http://www.ibm.com/support/knowledgecenter/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/apiref/c_restapi_oview.html).

Configuring license tracking

License tracking is enabled by default. Read the following topics to learn how you can configure license tracking. For more information about license tracking, see License tracking (../../../administering-apps/license-tracking).

- Configuring license tracking for client device and addressable device
- Configuring IBM License Metric Tool log files

Configuring license tracking for client device and addressable device

License tracking for client devices and addressable device is enabled by default. License reports are available in the MobileFirst Operations Console. You can specify the following JNDI properties to change the default settings for license tracking.

> **Note:** If you have a contract that defines the use of token licensing, see alsoInstalling and configuring for token licensing (../token-licensing).

You can specify the following JNDI properties to change the default settings for license tracking.

mfp.device.decommission.when
The number of days of inactivity after which a device is decommissioned by the device decommissioning task. License reports do not count decommissioned devices as active devices. The default value for the property is 90 days. Do not set a value lower than 30 days if your software is licensed by Client Device or by Addressable Device, or license reports might not be sufficient to prove compliance.

mfp.device.archiveDecommissioned.when
A value, in days, that defines when decommissioned devices are placed in an archive file when the decommissioning task is run. The archived devices are written to a file in the IBM MobileFirst Server **home\devices_archive** directory. The name of the file contains the time stamp when the archive file is created. The default value is 90 days.

mfp.device.decommissionProcessingInterval
Defines how often (in seconds) the decommissioning task is run. Default: 86400, which is one day. The decommissioning task performs the following actions:

- Decommissions inactive devices, based on the **mfp.device.decommission.when** setting.
- Optionally, archives older decommissioned devices, based on the **mfp.device.archiveDecommissioned.when** setting.
- Generates the license tracking report.

mfp.licenseTracking.enabled
A value that is used to enable or disable license tracking in IBM MobileFirst Foundation. By default, license tracking is enabled. For performance reasons, you can disable this flag when IBM MobileFirst Foundation is not licensed by Client Device or by Addressable Device. When device tracking is disabled, the license reports are also disabled and no license metrics are generated. In that case, only IBM License Metric Tool records for Application count are generated.

For more information about specifying JNDI properties, see List of JNDI properties for MobileFirst runtime.

Configuring IBM License Metric Tool log files

IBM MobileFirst Foundation generates IBM Software License Metric Tag (SLMT) files. Versions of IBM License Metric Tool that support IBM Software License Metric Tag can generate License Consumption Reports. Read this to understand how to configure the location and the maximum size of the generated files.

By default, the IBM Software License Metric Tag files are in the following directories:

- On Windows: **%ProgramFiles%\ibm\common\slm**
- On UNIX and UNIX-like operating systems: **/var/ibm/common/slm**

If the directories are not writable, the files are created in the log directory of the application server that runs the MobileFirst runtime environment.

You can configure the location and management of those files with the following properties:

- **license.metric.logger.output.dir**: Location of the IBM Software License Metric Tag files
- **license.metric.logger.file.size**: Maximum size of an SLMT file before a rotation is performed. The default size is 1 MB.
- **license.metric.logger.file.number**: Maximum number of SLMT archive files to keep in rotations. The default number is 10.

To change the default values, you must create a Java property file, with the format **key=value**, and provide the path to the properties file through the **license*metric*logger_configuration** JVM property.

For more information about IBM License Metric Tool reports, see Integration with IBM License Metric Tool (../../../administering-apps/license-tracking/#integration-with-ibm-license-metric-tool).

WebSphere Application Server SSL configuration and HTTP adapters

By setting a property, you can let HTTP adapters benefit from WebSphere SSL configuration.

By default, HTTP adapters do not use WebSphere SSL by concatenating the Java Runtime Environment (JRE) truststore with the MobileFirst Server keystore, which is described in Configuring the MobileFirst Server keystore (../../../authentication-and-security/configuring-the-mobilefirst-server-keystore). Also see Configuring SSL between MobileFirst adapters and back-end servers by using self-signed certificates (../../../administering-apps/deployment/#configuring-ssl-between-mobilefirst-adapters-and-back-end-servers-by-using-self-signed-certificates).

To have HTTP adapters use the WebSphere SSL configuration, set the **ssl.websphere.config** JNDI property to true. The setting has the following effects in order of precedence:

1. Adapters running on WebSphere use the WebSphere keystore and not the MobileFirst Server keystore.
2. If the **ssl.websphere.alias** property is set, the adapter uses the SSL configuration that is associated with the alias as set in this property.

Last modified on