Chapter 1. Installing Red Hat JBoss A-MQ as a Service

Red Hat JBoss A-MQ can generate a service wrapper that can be easily configured to install Red Hat JBoss A-MQ as a system service.

Installing Red Hat JBoss A-MQ as a system service is a three step process:

1. Generate the service wrapper for your system.

   See "Generating the Wrapper" on page 2.

2. Configure the service wrapper for your system.

   See "Configuring the Wrapper" on page 4.

3. Install the service wrapper as system service.

   See "Installing and Starting the Service" on page 8.

Generating the Wrapper

Overview

The Red Hat JBoss A-MQ console's `wrapper` feature generates a wrapper around the JBoss A-MQ runtime that allows you to install a message broker as a system service. The `wrapper` feature does not come preinstalled in the console, so before you can generate the service wrapper you must install the `wrapper` feature.

Once the feature is installed the console gains a **wrapper:install** command. Running this command generates a generic service wrapper in the JBoss A-MQ installation.

Procedure

To generate the service wrapper:

1. Start JBoss A-MQ in console mode using the **amq** command.
2. Once the console is started and the command prompt appears, enter **`features:install wrapper`**.

   The **features:install** command will locate the required libraries to provision the wrapper feature and deploy it into the run time.
3. Generate the wrapper by entering **`wrapper:install -n`** ***`serviceName`*** **`-d`** ***`displayName`*** **`-D`** ***`description`***.

   The **wrapper:install** command has the options described in Table 1.1 on page 2.

Table 1.1. Wrapper Install Options

Option	Default	Description
`-s`	`AUTO_START`	Specifies the mode in which the service is installed. Valid values are `AUTO_START` or `DEMAND_START`.
`-n`	`karaf`	Specifies the service name that will be used when installing the service.
`-d`		Specifies the display name of the service.
`-D`		Specifies the description of the service.

Generated files

The following files are generated and make up the service wrapper:

- `etc\`*`ServiceName`*`-wrapper(.exe)`—the executable file for the wrapper.
- `bin\`*`ServiceName`*`-service(.bat)`—the script used to install and remove the service.
- `etc\`*`ServiceName`*`-wrapper.conf`—the wrapper's configuration file.
- Three library files required by the service wrapper:
  - `lib\libwrapper.so`
  - `lib\karaf-wrapper.jar`
  - `lib\karaf-wrapper-main.jar`

Important

The only generated file you should modify is the configuration file.

Configuring the Wrapper

Overview

The service wrapper is configured by the *ServiceName*-wrapper.conf file, which is located under the *InstallDir*/etc/ directory.

There are several settings you may want to change including:

- the default environment settings
- the properties passed to the JVM
- the classpath
- the JMX settings
- the logging settings

Specifying the Red Hat JBoss A-MQ's environment

A broker's environment is controlled by three environment variables:

- KARAF_HOME—the location of the Red Hat JBoss A-MQ install directory.
- KARAF_BASE—the root directory containing the configuration and OSGi data specific to the broker instance.

  The configuration for the broker instance is stored in the *KARAF_BASE*/conf directory. Other data relating to the OSGi runtime is also stored beneath the base directory.
- KARAF_DATA—the directory containing the logging and persistence data for the broker.

Example 1.1 on page 4 shows the default values.

Example 1.1. Default Environment Settings

```
set.default.KARAF_HOME=InstallDir
set.default.KARAF_BASE=InstallDir
set.default.KARAF_DATA=InstallDir\data
```

Passing parameters to the JVM

If you want to pass parameters to the JVM, you do so by setting wrapper properties using the form `wrapper.java.additional.<n>`. *<n>* is a sequence number that must be distinct for each parameter.

One of the most useful things you can do by passing additional parameters to the JVM is to set Java system properties. The syntax for setting a Java system property is `wrapper.java.additional.<n>=-DPropName=PropValue`.

Example 1.2 on page 5 shows the default Java properties.

Example 1.2. Default Java System Properties

```
# JVM
# note that n is the parameter number starting from 1.
wrapper.java.additional.1=-Dkaraf.home="%KARAF_HOME%"
wrapper.java.additional.2=-Dkaraf.base="%KARAF_BASE%"
wrapper.java.additional.3=-Dkaraf.data="%KARAF_DATA%"
wrapper.java.additional.4=-Dcom.sun.managment.jmxremote
wrapper.java.additional.5=-Dkaraf.startLocalConsole=false
wrapper.java.additional.6=-Dkaraf.startRemoteShell=true
wrapper.java.additional.7=-Djava.endorsed.dirs="%JAVA_HOME%/jre/lib/en
dorsed;%JAVA_HOME%/lib/endorsed;%KARAF_HOME%/lib/endorsed"
wrapper.java.additional.8=-
Djava.ext.dirs="%JAVA_HOME%/jre/lib/ext;%JAVA_HOME%/lib/ext;%KARAF_HOME%/lib/ext"
```

Adding classpath entries

You add classpath entries using the syntax `wrapper.java.classpath.<n>`. *<n>* is a sequence number that must be distinct for each classpath entry.

Example 1.3 on page 5 shows the default classpath entries.

Example 1.3. Default Wrapper Classpath

```
wrapper.java.classpath.1=%KARAF_BASE%/lib/karaf-wrapper.jar
wrapper.java.classpath.2=%KARAF_HOME%/lib/karaf.jar
wrapper.java.classpath.3=%KARAF_HOME%/lib/karaf-jaas-boot.jar
wrapper.java.classpath.4=%KARAF_BASE%/lib/karaf-wrapper-main.jar
```

JMX configuration

The default service wrapper configuration does not enable JMX. It does, however, include template properties for enabling JMX. To enable JMX:

1. Locate the line `# Uncomment to enable jmx`.

   There are three properties, shown in Example 1.4 on page 6, that are used to configure JMX.

   Example 1.4. Wrapper JMX Properties

   ```
   # Uncomment to enable jmx
   #wrapper.java.additional.n=-Dcom.sun.management.jmxremote.port=1616
   #wrapper.java.additional.n=-Dcom.sun.management.jmxremote.authenticate=false
   #wrapper.java.additional.n=-Dcom.sun.management.jmxremote.ssl=false
   ```

2. Remove the `#` from in front of each of the properties.

3. Replace the `n` in each property to a number that fits into the sequence of addition properties established in the configuration.

You can change the settings to use a different port or secure the JMX connection.

For more information about using JMX see ???.

Configuring logging

The wrapper's logging in configured using the properties described in Table 1.2 on page 6.

Table 1.2. Wrapper Logging Properties

Property	Description
`wrapper.console.format`	Specifies how the logging information sent to the console is formated. The format consists of the following tokens: • `L`—log level • `P`—prefix • `D`—thread name • `T`—time • `Z`—time in milliseconds • `U`—approximate uptime in seconds (based on internal tick counter) • `M`—message

Property	Description
`wrapper.console.loglevel`	Specifies the logging level displayed on the console.
`wrapper.logfile`	Specifies the file used to store the log.
`wrapper.logfile.format`	Specifies how the logging information sent to the log file is formated.
`wrapper.console.loglevel`	Specifies the logging level sent to the log file.
`wrapper.console.maxsize`	Specifies the maximum size, in bytes, that the log file can grow to before the log is archived. The default value of 0 disables log rolling.
`wrapper.console.maxfiles`	Specifies the maximum number of archived log files which will be allowed before old files are deleted. The default value of 0 implies no limit.
`wrapper.syslog.loglevel`	Specifies the logging level for the sys/event log output.

For more information about Red Hat JBoss A-MQ logging see ???.

Installing and Starting the Service

Overview

The operating system determines the exact steps using to complete the installation of Red Hat JBoss A-MQ as a service. The **wrapper:install** command provides basic instructions for your operating system.

Windows

To install the service run ***InstallDir*\bin*ServiceName*-service.bat install**. If you used the default start setting, the service will start when Windows is launched. If you specified `DEMAND_START`, you will need to start the service manually.

To start the service manually run **net start "*ServiceName*"**. You can also use the Windows service UI.

To manually stop the service run **net stop "*ServiceName*"** You can also use the Windows service UI.

You remove the installed the service by running ***InstallDir*\bin*ServiceName*-service.bat remove**.

Redhat Linux

To install the service and configure it to start when the machine boots, run the following commands:

```
# ln -s InstallDir\bin\ServiceName-service /etc/init.d/
# chkconfig ServiceName-service --add
# chkconfig ServiceName-service on
```

To start the service manually run **service *ServiceName*-service start**.

To manually stop the service run **service *ServiceName*-service stop**.

You remove the installed the service by running the following commands:

```
#service ServiceName-service stop
# chkconfig ServiceName-service --del
# rm /etc/init.d/ServiceName-service
```

Ubuntu Linux

To install the service and configure it to start when the machine boots, run the following commands:

```
# ln -s InstallDir\bin\ServiceName-service /etc/init.d/
# update-rc.d ServiceName-service defaults
```

To start the service manually run **`/etc/init.d/`*`ServiceName`*`-service start`**.

To manually stop the service run **`/etc/init.d/`*`ServiceName`*`-service stop`**.

You remove the installed the service by running the following commands:

```
#/etc/init.d/ServiceName-service stop
# rm /etc/init.d/ServiceName-service
```