# Red Hat Fuse 7.2

## Installing on Apache Karaf

Installing Red Hat Fuse on the Apache Karaf container

Last Updated: 2019-01-05

# Red Hat Fuse 7.2 Installing on Apache Karaf

Installing Red Hat Fuse on the Apache Karaf container

## Legal Notice



## Abstract

It is easy to install Red Hat Fuse and tailor the installation to a particular environment.

# Table of Contents

**CHAPTER 1. INSTALLATION PREREQUISITES** . . . . . **3**
SUPPORTED PLATFORMS 3
JAVA RUNTIME 3
SUPPORTED STANDARDS 3
HARDWARE REQUIREMENTS 3
RECOMMENDED SOFTWARE 3

**CHAPTER 2. INSTALLATION TYPES** . . . . . **4**

**CHAPTER 3. INSTALLING ON APACHE KARAF** . . . . . **5**
GETTING THE ARCHIVE 5
UNPACKING THE ARCHIVE 5
USING THE IBM JDK 5

**CHAPTER 4. ADDING A REMOTE CONSOLE USER** . . . . . **6**

**CHAPTER 5. OFFLINE MODE** . . . . . **7**
5.1. DOWNLOAD REPOSITORY FOR MAVEN PROJECT 7
5.2. ENABLE OFFLINE MODE 7

**APPENDIX A. INSTALLING THE APACHE COMPONENTS** . . . . . **9**
A.1. GETTING THE EXTRAS ARCHIVE 9
A.2. CONTENTS OF THE EXTRAS ARCHIVE 9

**APPENDIX B. PREPARING TO USE MAVEN** . . . . . **10**
B.1. OVERVIEW 10
B.2. PREREQUISITES 10
B.3. ADDING THE RED HAT MAVEN REPOSITORIES 10
B.4. ARTIFACTS 12
B.5. MAVEN COORDINATES 12

# CHAPTER 1. INSTALLATION PREREQUISITES

Before attempting to install and use Fuse on Apache Karaf, make sure your system meets the minimum requirements.

## SUPPORTED PLATFORMS

Red Hat tests and supports Fuse products in the configurations listed at Red Hat JBoss Fuse Supported Configurations.

## JAVA RUNTIME

For details of the Java runtimes supported by Fuse on Apache Karaf, see Red Hat JBoss Fuse Supported Configurations.

**WARNING**

Do **not** install the Java runtime under a directory path that includes whitespace. For example, `C:\Program Files\Java\jdk8` is not an acceptable install path and will lead to unpredictable errors in Fuse on Apache Karaf at run time.

## SUPPORTED STANDARDS

Fuse on Apache Karaf supports the standards and protocols listed at Red Hat JBoss Fuse Supported Standards.

## HARDWARE REQUIREMENTS

The minimum hardware requirements for installing a full installation of Fuse on Apache Karaf are:

- 250 MB of free disk space
- 2 GB of RAM

In addition to the disk space required for the base installation, a running system will require space for caching, persistent message stores, and other functions.

## RECOMMENDED SOFTWARE

It is recommended that you use Maven with Fuse on Apache Karaf projects. For information about preparing to use Maven, see Appendix B, *Preparing to use Maven*.

# CHAPTER 2. INSTALLATION TYPES

The standard install package for Fuse 7.2 on Karaf is available for download from the Red Hat Customer Portal. It installs the standard assembly of the Apache Karaf container, including the full Fuse technology stack.

It is possible to create your own **custom assembly** of Fuse 7.2, containing a customized subset of the Fuse features and bundles.

# CHAPTER 3. INSTALLING ON APACHE KARAF

Red Hat Fuse is installed by unpacking an archive file on your file system.

## GETTING THE ARCHIVE

You can download the Fuse on Karaf archive from the Red Hat Customer Portal→Downloads→Red Hat JBoss Middleware→Downloads page, after you register and log in to your customer account.

Once logged in:

1. Select **Fuse**, listed under **Integration Platforms**, in the sidebar menu.
2. Select **7.2.0** from the **Version** drop-down list on the **Software Downloads** page.
3. Click the **Download** button next to the *Red Hat Fuse 7.2.0 on Karaf Installer*file.

## UNPACKING THE ARCHIVE

Fuse on Karaf is packaged as a **.zip** file. Using a suitable archive tool, such as **Zip**, unpack Fuse on Karaf into a directory to which you have full access.

**WARNING**

Do not unpack the archive file into a folder that has spaces in its path name. For example, do not unpack into **C:\Documents and Settings\Greco Roman\Desktop\fuse**.

**WARNING**

Do not unpack the archive file into a folder that has any of the following special characters in its path name: **#**, **%**, **^**, **"**.

## USING THE IBM JDK

If you are using the IBM JDK, remove the **saaj-api** jar from the ***installDir*/lib/endorsed** library using the following command:

```
rm lib/endorsed/org.apache.servicemix.specs.saaj-api-1.3-2.7.0.jar
```

Before invoking the **./bin/fuse** script:, set the **JAVA_OPTS** environment variable as follows:

```
JAVA_OPTS=-Xshareclasses:none
```

# CHAPTER 4. ADDING A REMOTE CONSOLE USER

Fuse on Karaf is not installed with a default user for the remote console. You must add a user before you can connect to the server's remote console. To add a user, edit **`InstallDir/etc/users.properties`**.

**IMPORTANT**

The information in this file is unencrypted so it is not suitable for environments that require strict security.

To add a user:

1. Open **`InstallDir/etc/users.properties`** in your favorite text editor.
2. Locate the following lines:

```
#admin = admin,_g_:admingroup
#_g_\:admingroup = group,admin,manager,viewer,systembundles,ssh
```

   Note that the first line has the syntax **`USER=PASSWORD,_g_:GROUP,ROLE1,ROLE2,...`**. In this example, the first line specifies a user, **`admin`**, with the password, **`admin`**, and the role group, **`admingroup`**.

3. Uncomment both lines by removing the leading # character.
4. Change the first **`admin`** to the desired user name.
5. Change the second **`admin`** to the desired password.
6. Save the changes.

# CHAPTER 5. OFFLINE MODE

You can run the Apache Karaf container in offline mode (that is, without an Internet connection). But if you are planning to deploy custom applications to the container, it might be necessary to download additional dependencies to a local Maven repository before you can run the container in offline mode with these applications.

To run the Apache Karaf container in offline mode, it is necessary to distinguish between the following kinds of dependency:

- **Runtime dependencies** — the dependencies required to run the Apache Karaf container, in its default configuration.
- **Build-time dependencies** — the dependencies required to build a custom application (which might include third-party libraries).

Here is a summary of what can be done in offline mode and what needs to be done in online mode (with an Internet connection):

- **Running the Apache Karaf container in its default configuration** — is supported in offline mode. The default configuration of the Apache Karaf container is specified by the **`featuresBoot`** property in the **`etc/org.apache.karaf.features.cfg`** file. The requisite dependencies are all provided in the **`system/`** sub-directory of the installation.
- **Installing additional features** — is, in general, **not** supported in offline mode. In principle, you can use the **`features:install`** command to install any of the features from the standard feature repositories (as specified by the **`featuresRepositories`** property in the **`etc/org.apache.karaf.features.cfg`** file), but the majority of these features must be downloaded from the Internet and are thus not supported in offline mode.
- **Deploying custom applications** — is, in general, **not** supported in offline mode. There may be some cases where an application with a minimal set of build-time dependencies is deployable offline, but in general, custom applications would have third-party dependencies that require an Internet connection (so that JAR files can be downloaded by Apache Maven).

If you do need to deploy an application with dependencies that are not available offline, you can use the Maven dependency plug-in to download the application's dependencies into a Maven offline repository. This customized Maven offline repository can then be distributed internally to any machines that do not have an Internet connection.

## 5.1. DOWNLOAD REPOSITORY FOR MAVEN PROJECT

From the project directory that contains the **`pom.xml`** file, run the following Maven command:

```
mvn org.apache.maven.plugins:maven-dependency-plugin:3.1.0:go-offline -
Dmaven.repo.local=/tmp/foo
```

All the Maven dependencies and plug-ins required to build the project will be downloaded to the **`/tmp/foo`** directory.

## 5.2. ENABLE OFFLINE MODE

To enable offline mode you must edit **`etc/org.ops4j.pax.url.mvn.cfg`**. Find the setting **`org.ops4j.pax.url.mvn.offline`** and replace the default **`false`** with **`true`**.

```
##
# If set to true, no remote repository will be accessed when resolving
artifacts
#
org.ops4j.pax.url.mvn.offline = true
```

# APPENDIX A. INSTALLING THE APACHE COMPONENTS

Red Hat Fuse provides an additional package to download, which contains the standard distributions of Apache Camel and Apache CXF. If you want to use a standard distribution of Apache Camel or Apache CXF (without the OSGi container) use the archived versions in the downloaded extras package.

## A.1. GETTING THE EXTRAS ARCHIVE

You can download the extras archive from the Red Hat Customer Portal→Downloads→Red Hat JBoss Middleware→Downloads page, after you register and log in to your customer account.

Once logged in:

1. Select **Fuse**, listed under **Integration Platforms**, in the sidebar menu.
2. Select **7.2.0** from the **Version** drop-down list on the **Software Downloads** page.
3. Download **fuse-extras-7.2.0.fuse-720050-redhat-00001.zip** archive.

## A.2. CONTENTS OF THE EXTRAS ARCHIVE

The extras archive file contains the following archive files nested inside it:

- **apache-camel-2.21.0.fuse-720050-redhat-00001.zip**
- **apache-cxf-3.1.11.fuse-720057-redhat-00001.zip**

You can copy these files to the desired location and decompress them using the appropriate utility for your platform.

**WARNING**

Do not unpack an archive file into a folder that has spaces in its path name. For example, do not unpack into **C:\Documents and Settings\Greco Roman\Desktop\fuse**.

# APPENDIX B. PREPARING TO USE MAVEN

## B.1. OVERVIEW

This section gives a brief overview of how to prepare Maven for building Red Hat Fuse projects and introduces the concept of Maven coordinates, which are used to locate Maven artifacts.

## B.2. PREREQUISITES

In order to build a project using Maven, you must have the following prerequisites:

- **Maven installation** — Maven is a free, open source build tool from Apache. You can download the latest version from the Maven download page.
- **Network connection** — whilst performing a build, Maven dynamically searches external repositories and downloads the required artifacts on the fly. By default, Maven looks for repositories that are accessed over the Internet. You can change this behavior so that Maven will prefer searching repositories that are on a local network.

**NOTE**

Maven can run in an offline mode. In offline mode Maven only looks for artifacts in its local repository.

## B.3. ADDING THE RED HAT MAVEN REPOSITORIES

In order to access artifacts from the Red Hat Maven repositories, you need to add them to Maven's **`settings.xml`** file. Maven looks for your **`settings.xml`** file in the **`.m2`** directory of the user's home directory. If there is not a user specified **`settings.xml`** file, Maven will use the system-level **`settings.xml`** file at **`M2_HOME/conf/settings.xml`**.

To add the Red Hat repositories to Maven's list of repositories, you can either create a new **`.m2/settings.xml`** file or modify the system-level settings. In the **`settings.xml`** file, add **`repository`** elements for the Red Hat repositories as shown in Adding the Red Hat Fuse Repositories to Maven.

**Adding the Red Hat Fuse Repositories to Maven**

```
<?xml version="1.0"?>
<settings>

  <profiles>
    <profile>
      <id>extra-repos</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      <repositories>
       <repository>
            <id>redhat-ga-repository</id>
            <url>https://maven.repository.redhat.com/ga</url>
            <releases>
                <enabled>true</enabled>
```

```
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>redhat-ea-repository</id>

<url>https://maven.repository.redhat.com/earlyaccess/all</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
        <repository>
          <id>jboss-public</id>
          <name>JBoss Public Repository Group</name>

<url>https://repository.jboss.org/nexus/content/groups/public/</url>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
            <id>redhat-ga-repository</id>
            <url>https://maven.repository.redhat.com/ga</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </pluginRepository>
        <pluginRepository>
            <id>redhat-ea-repository</id>

<url>https://maven.repository.redhat.com/earlyaccess/all</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </pluginRepository>
        <pluginRepository>
          <id>jboss-public</id>
          <name>JBoss Public Repository Group</name>

<url>https://repository.jboss.org/nexus/content/groups/public</url>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>

  <activeProfiles>
```

```
    <activeProfile>extra-repos</activeProfile>
  </activeProfiles>

</settings>
```

## B.4. ARTIFACTS

The basic building block in the Maven build system is an *artifact*. The output of an artifact, after performing a Maven build, is typically an archive, such as a JAR or a WAR.

## B.5. MAVEN COORDINATES

A key aspect of Maven functionality is the ability to locate artifacts and manage the dependencies between them. Maven defines the location of an artifact using the system of *Maven coordinates*, which uniquely define the location of a particular artifact. A basic coordinate tuple has the form, **{*groupId*, *artifactId*, *version*}**. Sometimes Maven augments the basic set of coordinates with the additional coordinates, *packaging* and *classifier*. A tuple can be written with the basic coordinates, or with the additional *packaging* coordinate, or with the addition of both the *packaging* and *classifier* coordinates, as follows:

```
groupId:artifactId:version
groupId:artifactId:packaging:version
groupId:artifactId:packaging:classifier:version
```

Each coordinate can be explained as follows:

***groupdId***

Defines a scope for the name of the artifact. You would typically use all or part of a package name as a group ID — for example, **`org.fusesource.example`**.

***artifactId***

Defines the artifact name (relative to the group ID).

***version***

Specifies the artifact's version. A version number can have up to four parts: **`n.n.n.n`**, where the last part of the version number can contain non-numeric characters (for example, the last part of **`1.0-SNAPSHOT`** is the alphanumeric substring, **`0-SNAPSHOT`**).

***packaging***

Defines the packaged entity that is produced when you build the project. For OSGi projects, the packaging is **`bundle`**. The default value is **`jar`**.

***classifier***

Enables you to distinguish between artifacts that were built from the same POM, but have different content.

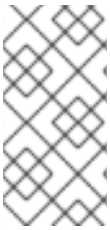
The group ID, artifact ID, packaging, and version are defined by the corresponding elements in an artifact's POM file. For example:

```
<project ... >
  ...
  <groupId>org.fusesource.example</groupId>
  <artifactId>bundle-demo</artifactId>
  <packaging>bundle</packaging>
  <version>1.0-SNAPSHOT</version>
```

```
  ...
</project>
```

For example, to define a dependency on the preceding artifact, you could add the following **dependency** element to a POM:

```
<project ... >
  ...
  <dependencies>
    <dependency>
      <groupId>org.fusesource.example</groupId>
      <artifactId>bundle-demo</artifactId>
      <version>1.0-SNAPSHOT</version>
    </dependency>
  </dependencies>
  ...
</project>
```



**NOTE**

It is **not** necessary to specify the **`bundle`** package type in the preceding dependency, because a bundle is just a particular kind of JAR file and **`jar`** is the default Maven package type. If you do need to specify the packaging type explicitly in a dependency, however, you can use the **`type`** element.