Red Hat JBoss Data Grid 6.2

Getting Started Guide

For use with Red Hat JBoss Data Grid 6.2.1

Edition 3

Last Updated: 2017-11-21

Red Hat JBoss Data Grid 6.2 Getting Started Guide

For use with Red Hat JBoss Data Grid 6.2.1
Edition 3

Misha Husnain Ali
Red Hat Engineering Content Services
mhusnain@redhat.com

Gemma Sheldon
Red Hat Engineering Content Services
gsheldon@redhat.com

Mandar Joshi
Red Hat Engineering Content Services
majoshi@redhat.com

Rakesh Ghatvisave
Red Hat Engineering Content Services
rghatvis@redhat.com

Legal Notice

Abstract

This guide outlines introductory concepts and operations within Red Hat JBoss Data Grid 6.2.1

Table of Contents

PART I. INTRODUCING RED HAT JBOSS DATA GRID

CHAPTER 1. RED HAT JBOSS DATA GRID

Red Hat JBoss Data Grid is a distributed in-memory data grid, which provides the following capabilities:

- Schemaless key-value store – JBoss Data Grid is a NoSQL database that provides the flexibility to store different objects without a fixed data model.
- Grid-based data storage – JBoss Data Grid is designed to easily replicate data across multiple nodes.
- Elastic scaling – Adding and removing nodes is simple and non-disruptive.
- Multiple access protocols – It is easy to access the data grid using REST, Memcached, Hot Rod, or simple map-like API.

JBoss Data Grid 6.2 is based on Infinispan version 6.0.

Report a bug

1.1. SUPPORTED CONFIGURATIONS

The set of supported features, configurations, and integrations for Red Hat JBoss Data Grid (current and past versions) are available at the Supported Configurations page at https://access.redhat.com/knowledge/articles/115883.

Report a bug

1.2. COMPONENTS AND VERSIONS

Red Hat JBoss Data Grid includes many components for Library and Remote Client-Server modes. A comprehensive (and up to date) list of components included in each of these usage modes and their versions is available in the *Red Hat JBoss Data Grid Component Details* page at https://access.redhat.com/site/articles/488833

Report a bug

1.3. RED HAT JBOSS DATA GRID USAGE MODES

Red Hat JBoss Data Grid offers two usage modes:

- Remote Client-Server mode
- Library mode

Report a bug

1.3.1. Remote Client-Server Mode

Remote Client-Server mode provides a managed, distributed, and clusterable data grid server. Applications can remotely access the data grid server using **Hot Rod**, **Memcached** or **REST** client APIs.

All Red Hat JBoss Data Grid operations in Remote Client-Server mode are non-transactional. As a result, a number of features cannot be performed when running JBoss Data Grid in Remote Client-Server mode.

There are a number of benefits to running JBoss Data Grid in Remote Client-Server mode if Library mode features are not required. Remote Client-Server mode is client language agnostic, provided there is a client library for your chosen protocol. As a result, Remote Client-Server mode provides:

- easier scaling of the data grid.
- easier upgrades of the data grid without impact on client applications.

Run the following commands to start JBoss Data Grid in Remote Client-Server mode.

For Linux:

```
$JBOSS_HOME/bin/standalone.sh
```

For Windows:

```
$JBOSS_HOME\bin\standalone.bat
```

Report a bug

1.3.2. Library Mode

Library mode allows building and deploying a custom runtime environment. The Library mode hosts a single data grid node in the applications process, with remote access to nodes hosted in other JVMs. Tested containers for Red Hat JBoss Data Grid 6 Library mode includes Red Hat JBoss Web Server 2 and JBoss Enterprise Application Platform 6.

A number of features in JBoss Data Grid can be used in Library mode, but not Remote Client-Server mode.

The following features require Library mode:

- transactions
- listeners and notifications

JBoss Data Grid can also be run as a standalone application in Java SE. Standalone mode is a supported alternative to running JBoss Data Grid in a container.

Report a bug

1.4. RED HAT JBOSS DATA GRID BENEFITS

Red Hat JBoss Data Grid provides the following benefits:

Benefits of JBoss Data Grid

Performance

Accessing objects from local memory is faster than accessing objects from remote data stores (such as a database). JBoss Data Grid provides an efficient way to store in-memory objects coming from a slower data source, resulting in faster performance than a remote data store. JBoss Data Grid also offers optimization for both clustered and non clustered caches to further improve performance.

Consistency

Storing data in a cache carries the inherent risk: at the time it is accessed, the data may be outdated (stale). To address this risk, JBoss Data Grid uses mechanisms such as cache invalidation and expiration to remove stale data entries from the cache. Additionally, JBoss Data Grid supports JTA, distributed (XA) and two-phase commit transactions along with transaction recovery and a version API to remove or replace data according to saved versions.

Massive Heap and High Availability

In JBoss Data Grid, applications no longer need to delegate the majority of their data lookup processes to a large single server database for performance benefits. JBoss Data Grid employs techniques such as replication and distribution to completely remove the bottleneck that exists in the majority of current enterprise applications.

> **Example 1.1. Massive Heap and High Availability Example**
>
> In a sample grid with 16 blade servers, each node has 2 GB storage space dedicated for a replicated cache. In this case, all the data in the grid is copies of the 2 GB data. In contrast, using a distributed grid (assuming the requirement of one copy per data item, resulting in the capacity of the overall heap being divided by two) the resulting memory backed virtual heap contains 16 GB data. This data can now be effectively accessed from anywhere in the grid. In case of a server failure, the grid promptly creates new copies of the lost data and places them on operational servers in the grid.

Scalability

A significant benefit of a distributed data grid over a replicated clustered cache is that a data grid is scalable in terms of both capacity and performance. Add a node to JBoss Data Grid to increase throughput and capacity for the entire grid. JBoss Data Grid uses a consistent hashing algorithm that limits the impact of adding or removing a node to a subset of the nodes instead of every node in the grid.

Due to the even distribution of data in JBoss Data Grid, the only upper limit for the size of the grid is the group communication on the network. The network's group communication is minimal and restricted only to the discovery of new nodes. Nodes are permitted by all data access patterns to communicate directly via peer-to-peer connections, facilitating further improved scalability.
JBoss Data Grid clusters can be scaled up or down in real time without requiring an infrastructure restart. The result of the real time application of changes in scaling policies results in an exceptionally flexible environment.

Data Distribution

JBoss Data Grid uses consistent hash algorithms to determine the locations for keys in clusters. Benefits associated with consistent hashing include:

- cost effectiveness.
- speed.
- deterministic location of keys with no requirements for further metadata or network traffic.

Data distribution ensures that sufficient copies exist within the cluster to provide durability and fault tolerance, while not an abundance of copies, which would reduce the environment's scalability.

Persistence

JBoss Data Grid exposes a **`CacheStore`** interface and several high-performance implementations, including the JDBC Cache stores and file system based cache stores. Cache stores can be used to

populate the cache when it starts and to ensure that the relevant data remains safe from corruption. The cache store also overflows data to the disk when required to prevent running out of memory.

Language bindings

JBoss Data Grid supports both the popular Memcached protocol, with existing clients for a large number of popular programming languages, as well as an optimized JBoss Data Grid specific protocol called Hot Rod. As a result, instead of being restricted to Java, JBoss Data Grid can be used for any major website or application. Additionally, remote caches can be accessed using the HTTP protocol via a RESTful API.

Management

In a grid environment of several hundred or more servers, management is an important feature. JBoss Operations Network, the enterprise network management software, is the best tool to manage multiple JBoss Data Grid instances. JBoss Operations Network's features allow easy and effective monitoring of the Cache Manager and cache instances.

Remote Data Grids

Rather than scale up the entire application server architecture to scale up your data grid, JBoss Data Grid provides a Remote Client-Server mode which allows the data grid infrastructure to be upgraded independently from the application server architecture. Additionally, the data grid server can be assigned different resources than the application server and also allow independent data grid upgrades and application redeployment within the data grid.

Report a bug

1.5. RED HAT JBOSS DATA GRID VERSION INFORMATION

Red Hat JBoss Data Grid is based on Infinispan, the open source community version of the data grid software. Infinispan uses code, designs, and ideas from JBoss Cache, which have been tried, tested, and proved in high stress environments. As a result, JBoss Data Grid's first release is version 6.0 as a result of its deployment history.

The following table lists the correlation between JBoss Data Grid and Infinispan versions.

Table 1.1. JBoss Data Grid and Infinispan Correlation

JBoss Data Grid Product	Infinispan Version
JBoss Data Grid 6.0.0	Infinispan 5.1.5
JBoss Data Grid 6.0.1	Infinispan 5.1.7
JBoss Data Grid 6.1.0	Infinispan 5.2.4
JBoss Data Grid 6.2.0	Infinispan 6.0.1

Report a bug

1.6. RED HAT JBOSS DATA GRID CACHE ARCHITECTURE

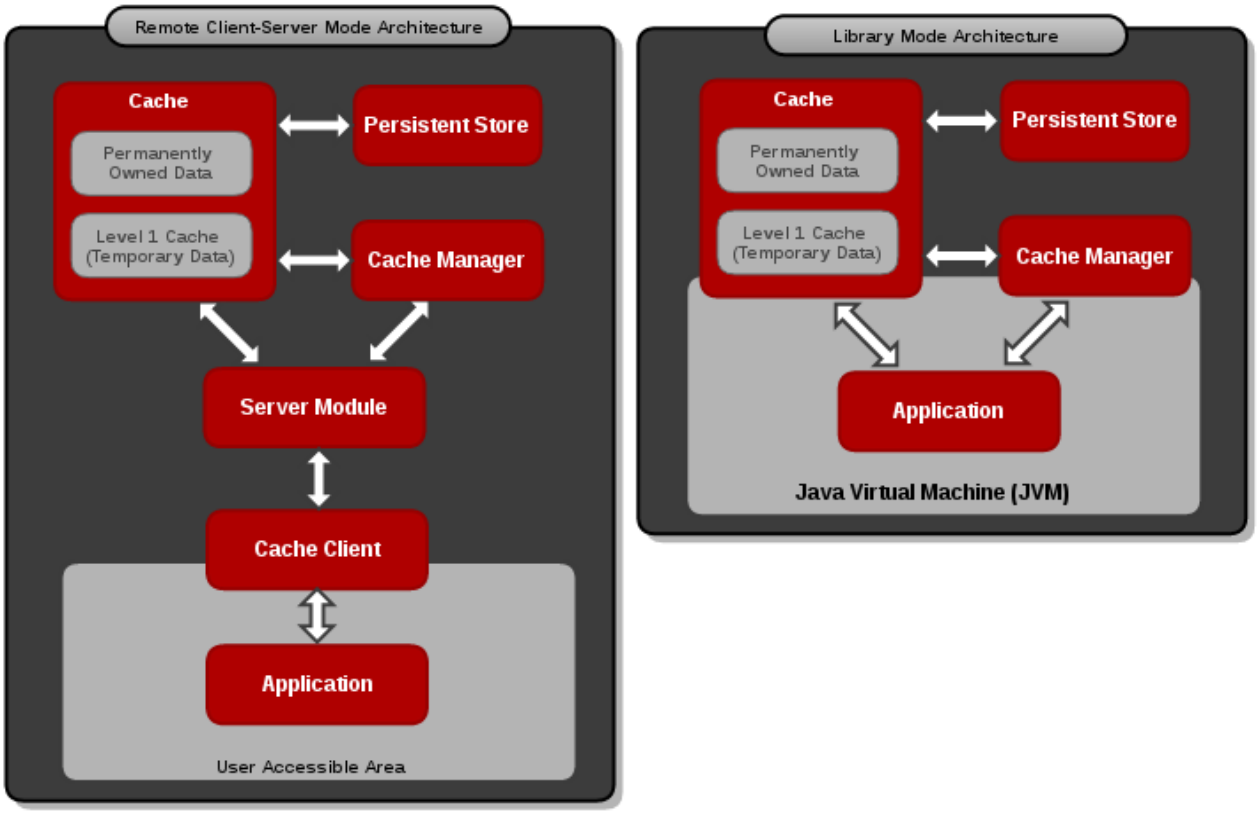


Figure 1.1. JBoss Data Grid Cache Architecture

Red Hat JBoss Data Grid's cache infrastructure depicts the individual elements and their interaction with each other in each JBoss Data Grid Usage Mode (Library and Remote Client-Server). For clarity, each cache architecture diagram is separated into two parts:

- Elements that a user cannot directly interact with (depicted within a dark box). In Remote Client-Server mode, this includes Persistent Store, Cache, Cache Manager, L1 Cache, and Server Module. In Library mode, user cannot directly interact with Persistent Store and L1 Cache.
- Elements that a user can interact directly with (depicted within a light gray box). In Remote Client-Server mode, this includes the Application and the Cache Client. In Library mode, programmatic configurations are used to allow the user to interact with the Cache and Cache Manager, as well as the Application.

Cache Architecture Elements

JBoss Data Grid's cache architecture includes the following elements:

1. The Persistent Store is an optional component. It can permanently store the cached entries for restoration after a data grid shutdown.
2. The Level 1 Cache (or L1 Cache) stores remote cache entries after they are initially accessed, preventing unnecessary remote fetch operations for each subsequent use of the same entries.
3. The Cache Manager controls the life cycle of Cache instances and can store and retrieve them when required.
4. The Cache is the main component for storage and retrieval of the key-value entries.

Library and Remote Client-Server Mode Architecture

In Library mode, the Application (user code) can interact with the Cache and Cache Manager components directly. In this case the Application resides in the same Java Virtual Machine (JVM) and can call Cache and Cache Manager Java API methods directly.

In Remote Client-Server mode, the Application does not directly interact with the cache. Additionally, the Application usually resides in a different JVM, on different physical host, or does not need to be a Java Application. In this case, the Application uses a Cache Client that communicates with a remote JBoss Data Grid Server over the network using one of the supported protocols such as Memcached, Hot Rod, or REST. The appropriate server module handles the communication on the server side. When a request is sent to the server remotely, it translates the protocol back to the concrete operations performed on the cache component to store and retrieve data.

Report a bug

1.7. RED HAT JBOSS DATA GRID APIS

Red Hat JBoss Data Grid provides the following programmable APIs:

- Cache
- Batching
- Grouping
- Persistence (formerly CacheStore)
- ConfigurationBuilder
- Externalizable
- Notification (also known as the Listener API because it deals with Notifications and Listeners)

JBoss Data Grid offers the following APIs to interact with the data grid in Remote Client-Server mode:

- The Asynchronous API (can only be used in conjunction with the Hot Rod Client in Remote Client-Server Mode)
- The REST Interface
- The Memcached Interface
- The Hot Rod Interface
    - The RemoteCache API

Report a bug

PART II. DOWNLOAD AND INSTALL RED HAT JBOSS DATA GRID

CHAPTER 2. DOWNLOAD RED HAT JBOSS DATA GRID

2.1. RED HAT JBOSS DATA GRID INSTALLATION PREREQUISITES

The only prerequisite to set up Red Hat JBoss Data Grid is a Java Virtual Machine (compatible with Java 6.0 or later) installed.

Report a bug

2.2. JAVA VIRTUAL MACHINE

A Java Virtual Machine (JVM) is a virtual environment that runs and executes Java programs on a host operating system. The JVM acts as an abstract computer and is a platform-independent execution environment that converts the Java programming code into machine language. A Java Virtual Machine (JVM) makes Java portable by providing an abstraction layer between the compiled Java program and the underlying hardware platform and operating system.

Red Hat recommends using OpenJDK Java platform as it is an open source, supported Java Virtual Machine that runs well on Red Hat Enterprise Linux systems. For Windows users, Oracle JDK 1.6 installation is recommended.

Report a bug

2.3. INSTALL OPENJDK ON RED HAT ENTERPRISE LINUX

Procedure 2.1. Install OpenJDK on Red Hat Enterprise Linux

1. **Subscribe to the Base Channel**
   Obtain the OpenJDK from the RHN base channel. Your installation of Red Hat Enterprise Linux is subscribed to this channel by default.

2. **Install the Package**
   Use the yum utility to install OpenJDK:

   ```
   $ sudo yum install java-1.6.0-openjdk-devel
   ```

3. **Verify that OpenJDK is the System Default**
   Ensure that the correct JDK is set as the system default as follows:

   a. Log in as a user with root privileges and run the alternatives command:

   ```
   $ /usr/sbin/alternatives --config java
   ```

   b. Depending on the OpenJDK version, select **`/usr/lib/jvm/jre-1.6.0-openjdk.x86_64/bin/java`** **`/usr/lib/jvm/jre-1.7.0-openjdk.x86_64/bin/java`**.

   c. Use the following command to set **`javac`**:

   ```
   $ /usr/sbin/alternatives --config javac
   ```

d. Depending on the OpenJDK version used, select **`/usr/lib/jvm/java-1.6.0-openjdk/bin/java`** or **`/usr/lib/jvm/java-1.7.0-openjdk/bin/java`**.

Report a bug

2.4. DOWNLOAD AND INSTALL JBOSS DATA GRID

Use the following steps to download and install Red Hat JBoss Data Grid:

1. Download JBoss Data Grid from the Red Hat Customer Service Portal
2. Verify the downloaded files
3. Install JBoss Data Grid

Report a bug

2.4.1. Download Red Hat JBoss Data Grid

Follow the listed steps to download Red Hat JBoss Data Grid from the Customer Service Portal:

Procedure 2.2. Download JBoss Data Grid

1. **Access the Customer Service Portal**
   Log into the Customer Service Portal at https://access.redhat.com.
2. **Locate the Product**
   Mouse over **Downloads** and navigate to **JBoss Enterprise Middleware**.
3. **Select the Product**
   Select **Data Grid**.
4. **Download JBoss Data Grid**
   Select the appropriate JBoss Data Grid download and click the **Download** link.

   Select **JBoss Data Grid Server {VERSION}** for JBoss Data Grid with the Remote Client-Server mode or **JBoss Data Grid Library {VERSION}** for JBoss Data Grid with the Library mode.

Report a bug

2.4.2. About the Red Hat Customer Portal

The *Red Hat Customer Portal* is the centralized platform for Red Hat knowledge and subscription resources. Use the *Red Hat Customer Portal* to:

- Manage and maintain Red Hat entitlements and support contracts;
- Download officially-supported software;
- Access product documentation and the Red Hat Knowledgebase;
- Contact Global Support Services; and
- File bugs against Red Hat products.

The Customer Portal is available here: https://access.redhat.com.

Report a bug

2.4.3. Checksum Validation

Checksum validation is used to ensure a downloaded file has not been corrupted. Checksum validation employs algorithms that compute a fixed-size datum (or checksum) from an arbitrary block of digital data. If two parties compute a checksum of a particular file using the same algorithm, the results will be identical. Therefore, when computing the checksum of a downloaded file using the same algorithm as the supplier, if the checksums match, the integrity of the file is confirmed. If there is a discrepancy, the file has been corrupted in the download process.

Report a bug

2.4.4. Verify the Downloaded File

Procedure 2.3. Verify the Downloaded File

1. To verify that a file downloaded from the Red Hat Customer Portal is error-free, access the portal site and go to that package's **`Software Details`** page. The Software Details page displays the **`MD5`** and **`SHA256`** "checksum" values. Use the checksum values to check the integrity of the file.

2. Open a terminal window and run either the **`md5sum`** or **`sha256sum`** command, with the downloaded file as an argument. The program displays the checksum value for the file as the output for the command.

3. Compare the checksum value returned by the command to the corresponding value displayed on the **`Software Details`** page for the file.

NOTE

Microsoft Windows does not come equipped with a checksum tool. Windows operating system users have to download a third-party product instead.

Result

If the two checksum values are identical then the file has not been altered or corrupted and is, therefore, safe to use.

If the two checksum values are not identical, then download the file again. A difference between the checksum values means that the file has either been corrupted during download or has been modified since it was uploaded to the server. If, after several downloads, the checksum will still not successfully validate, contact Red Hat Support for assistance.

Report a bug

2.4.5. Install Red Hat JBoss Data Grid

Prerequisite

Locate the appropriate version, platform, and file type and download Red Hat JBoss Data Grid from the Customer Service Portal.

Procedure 2.4. Install JBoss Data Grid

1. Copy the downloaded JBoss Data Grid package to the preferred location on your machine.
2. Run the following command to unzip the downloaded JBoss Data Grid package:

   ```
   $ unzip JDG_PACKAGE
   ```

   Replace *JDG_PACKAGE* with the name of the JBoss Data Grid usage mode package downloaded from the Red Hat Customer Portal.
3. The resulting unzipped directory will now be referred to as *$JDG_HOME*.

Report a bug

2.4.6. Red Hat Documentation Site

Red Hat's official documentation site is available at https://access.redhat.com/site/documentation/. There you will find the latest version of every book, including this one.

Report a bug

CHAPTER 3. INSTALL AND USE THE MAVEN REPOSITORIES

3.1. ABOUT MAVEN

Apache Maven is a distributed build automation tool used in Java application development to create, manage, and build software projects. Maven uses standard configuration files called Project Object Model, or POM, files to define projects and manage the build process. POMs describe the module and component dependencies, build order, and targets for the resulting project packaging and output using an XML file. This ensures that the project is built in a correct and uniform manner.

IMPORTANT

Red Hat JBoss Data Grid requires Maven 3 (or better) for all quick starts and general use.

Visit the Maven Download page (http://maven.apache.org/download.html) for instructions for downloading and installing Maven.

Report a bug

3.2. REQUIRED MAVEN REPOSITORIES

Red Hat JBoss Data Grid Quickstarts require the following Maven repositories to be set up as a prerequisite:

- The JBoss Data Grid Maven Repository
- The JBoss Enterprise Application Platform Maven Repository

Both Maven repositories are installed in the same way. As a result, the subsequent instructions are for both repositories.

Report a bug

3.3. INSTALL THE MAVEN REPOSITORY

There are three ways to install the required repositories:

1. On your local file system.
2. On Apache Web Server.
3. With a Maven repository manager.

Use the option that best suits your environment.

Report a bug

3.3.1. Local File System Repository Installation

This option is best suited for initial testing in a small team. Follow the outlined procedure to extract the Red Hat JBoss Data Grid and JBoss Enterprisenbsp;Application Platform Maven repositories to a directory in your local file system:

Procedure 3.1. Local File System Repository Installation (JBoss Data Grid)

1. **Log Into the Customer Service Portal**
   In a browser window, navigate to the Customer Service Portal and log in.

2. **Download the JBoss Data Grid Repository File**
   Download the **`jboss-datagrid-{VERSION}-maven-repository.zip`** file from the Red Hat Customer Portal.

3. Unzip the file to a directory on your local file system such as **`$JDG_HOME/projects/maven-repositories/`**

Report a bug

3.3.2. Maven Repository Manager Installation

This option is ideal if you are already using a repository manager.

The Red Hat JBoss Data Grid and JBoss Enterprise Application Server repositories are installed using a Maven repository manager using its documentation. Examples of such repository managers are:

- Apache Archiva: http://archiva.apache.org/
- JFrog Artifactory: http://www.jfrog.com/products.php
- Sonatype Nexus: http://nexus.sonatype.org/ For details, see Section B.1, “Install the JBoss Enterprise Application Platform Repository Using Nexus”

Report a bug

3.4. CONFIGURE THE MAVEN REPOSITORY

To configure the installed Red Hat JBoss Data Grid Maven repository, edit the **`settings.xml`** file. The default version of this file is available in the **`conf`** directory of your Maven installation.

Maven user settings are located in the **`.m2`** sub-directory of the user's home directory. See http://maven.apache.org/settings.html (the Maven documentation) for more information about configuring Maven.

See Section B.2, “Maven Repository Configuration Example” to view a sample Maven configuration.

Report a bug

3.4.1. Next Steps

After the newest available version of Red Hat JBoss Data Grid is installed and Maven is set up and configured, see Chapter 6, *Create a New Red Hat JBoss Data Grid Project* to learn how to use JBoss Data Grid for the first time.

Report a bug

PART III. RUN RED HAT JBOSS DATA GRID IN REMOTE CLIENT-SERVER MODE

CHAPTER 4. RUN RED HAT JBOSS DATA GRID IN REMOTE CLIENT-SERVER MODE

4.1. PREREQUISITES

The following is a list of prerequisites to run Red Hat JBoss Data Grid in Remote Client-Server mode for the first time:

- Ensure an appropriate version of OpenJDK is installed. For more information, see Section 2.3, “Install OpenJDK on Red Hat Enterprise Linux”
- Download and install the latest version of JBoss Data Grid. For more information, see Section 2.4.1, “Download Red Hat JBoss Data Grid”

Report a bug

4.2. RUN RED HAT JBOSS DATA GRID IN STANDALONE MODE

Standalone mode refers to a single instance of Red Hat JBoss Data Grid operating in local mode. In local mode, JBoss Data Grid operates as a simple single-node in-memory data cache.

Run the following script to start JBoss Data Grid in standalone mode:

```
$JDG_HOME/bin/standalone.sh
```

This command starts JBoss Data Grid using the default configuration information provided in the **`$JDG_HOME/standalone/configuration/standalone.xml`** file.

Report a bug

4.3. RUN RED HAT JBOSS DATA GRID IN CLUSTERED MODE

Clustered mode refers to a cluster made up of two or more Red Hat JBoss Data Grid instances.

Run the following script to start JBoss Data Grid in clustered mode:

```
$JDG_HOME/bin/clustered.sh
```

This command starts JBoss Data Grid using the default configuration information provided in the **`$JDG_HOME/standalone/configuration/clustered.xml`** file.

Report a bug

4.4. RUN RED HAT JBOSS DATA GRID WITH A CUSTOM CONFIGURATION

To run Red Hat JBoss Data Grid with a custom configuration, add a file in the **`$JDG_HOME/standalone/configuration`** directory.

Use the following command to specify the created custom configuration file for standalone mode:

```
$JDG_HOME/bin/standalone.sh -c ${FILENAME}
```

Use the following command to specify the created custom configuration file for clustered mode:

```
$JDG_HOME/bin/clustered.sh -c ${FILENAME}
```

The ***-c*** used for this script does not allow absolute paths, therefore the specified file must be available in the **$JDG_HOME/standalone/configuration** directory.

If the command is run without the ***-c*** parameter, JBoss Data Grid uses the default configuration.

Report a bug

4.5. SET AN IP ADDRESS TO RUN RED HAT JBOSS DATA GRID

For production use, the Red Hat JBoss Data Grid server must be bound to a specified IP address rather than to **127.0.0.1/localhost**. Use the ***-b*** parameter with the script to specify an IP address.

For standalone mode, set the IP address as follows:

```
$JDG_HOME/bin/standalone.sh -b ${IP_ADDRESS}
```

For clustered mode, set the IP address as follows:

```
$JDG_HOME/bin/clustered.sh -b ${IP_ADDRESS}
```

Report a bug

4.6. RUNNING RED HAT JBOSS DATA GRID

JBoss Data Grid can be run in one of three ways:

- Use the following command to run JBoss Data Grid using the configuration defined in the **standalone.xml** file (located at **$JDG_HOME/standalone/configuration**):

  ```
  $JDG_HOME/bin/standalone.sh
  ```

- Use the following command with an appended **-c** followed by the configuration file name to run JBoss Data Grid with a non-default configuration file:

  ```
  $JDG_HOME/bin/standalone.sh -c clustered.xml
  ```

- Use the following command to run JBoss Data Grid with a clustered configuration:

  ```
  $JDG_HOME/bin/clustered.sh
  ```

Report a bug

CHAPTER 5. RUN A RED HAT JBOSS DATA GRID AS A NODE WITHOUT ENDPOINTS

To communicate with each other, services send messaging using channels. An endpoint is a communications point for services, used to send and receive messages from channels. As a result, a node with no endpoints can communicate with other nodes in the cluster, but not with clients.

Report a bug

5.1. BENEFITS OF A NODE WITHOUT ENDPOINTS

The primary benefit for creating a node without endpoints in Red Hat JBoss Data Grid involves data replication.

A node without any endpoints cannot be accessed by the client directly. As a result, they are used to replicate data from another node that can communicate with clients. The result is a node with a backup copy of the data that cannot be accessed by the client, which protects it from failure via an error sent by the client.

Report a bug

5.2. SAMPLE CONFIGURATION FOR A NODE WITHOUT ENDPOINTS

Red Hat JBoss Data Grid provides a sample configuration to configure a node without an endpoint. The following process outlines how to access this example:

Procedure 5.1. Find the JBoss Data Grid Sample Configuration for a Node Without Endpoints

1. **Extract the JBoss Data Grid ZIP**
   1. Extract the ZIP file for JBoss Data Grid Remote Client-Server mode. This is named **`jboss-datagrid-server-${version}-GA`**. Add the relevant version to the file name.

2. **Navigate to the Appropriate Folder**
   In the extracted folder, navigate to the **`$JDG_HOME/docs/examples/config`** folder.

3. **Find the Configuration Sample File**
   View the **`standalone-storage-only.xml`** file, which contains the configuration for a node with no endpoints.

Report a bug

5.3. CONFIGURE A NODE WITH NO ENDPOINTS

A standard configuration, such as a standalone high availability configuration, can be changed into a configuration for a node with no endpoints using the following steps:

- Remove the ***`datagrid`*** subsystem.
- Remove the ***`modcluster`*** subsystem.
- Remove the ***`datasource`*** definition.
- Remove ***`socket-bindings`*** for ***`mod_cluster`***, **`Hot Rod`** and **`memcached`**.

Removing the listed items ensure that all endpoints are removed from the configuration and that clustering is not possible. The resulting configuration is a node with no endpoints.

Report a bug

PART IV. RUN RED HAT JBOSS DATA GRID IN LIBRARY MODE

This part includes information about using Red Hat JBoss Data Grid in Library Mode.

- As a prerequisite for the subsequent chapters, set up a new project using the instructions in Chapter 6, *Create a New Red Hat JBoss Data Grid Project*
- Next, use JBoss Data Grid either as an embedded cache (see Chapter 7, *Run Red Hat JBoss Data Grid in Library Mode (Single-Node Setup)* for more information) or as a clustered cache (see Chapter 8, *Run Red Hat JBoss Data Grid in Library Mode (Multi-Node Setup)*. Each tutorial is based on an Infinispan quickstart.
- Finally, monitor Red Hat JBoss EAP applications using JBoss Data Grid using the instructions in Chapter 9, *Monitor Red Hat JBoss Data Grid Applications in Red Hat JBoss EAP*

Report a bug

CHAPTER 6. CREATE A NEW RED HAT JBOSS DATA GRID PROJECT

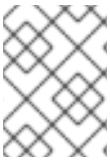
This chapter is a guide to creating a new Red Hat JBoss Data Grid project. The tasks prescribed are a prerequisite for the quickstart tasks provided in Chapter 7, *Run Red Hat JBoss Data Grid in Library Mode (Single-Node Setup)* and Chapter 8, *Run Red Hat JBoss Data Grid in Library Mode (Multi-Node Setup)*

Report a bug

6.1. ADD DEPENDENCIES TO YOUR PROJECT

Set up Red Hat JBoss Data Grid by adding dependencies to your project. If you are using Maven or other build systems that support Maven dependencies, add the following to your **pom.xml** file, located in the Maven repository folder:

```
<dependency>
          <groupId>org.infinispan</groupId>
          <artifactId>infinispan-core</artifactId>
          <version>5.1.5.FINAL-redhat-1</version>
</dependency>
```



NOTE

Replace the **version** value with the appropriate version of the libraries included in JBoss Data Grid.

Report a bug

6.2. ADD A PROFILE TO YOUR PROJECT

To enable the JBoss Maven repository for your project, add a profile to your **settings.xml** file in **$HOME/.m2/settings.xml** as follows:

```
<profile>
    <id>jboss-datagrid-repository</id>
    <repositories>
 <repository>
     <id>jboss-datagrid-repository</id>
     <name>JBoss Data Grid Maven Repository</name>
           <url>file:///path/to/repo/jboss-datagrid-{VERSION}-maven-
repository</url>
           <layout>default</layout>
           <releases>
            <enabled>true</enabled>
            <updatePolicy>never</updatePolicy>
           </releases>
           <snapshots>
            <enabled>false</enabled>
            <updatePolicy>never</updatePolicy>
           </snapshots>
 </repository>
```

```
 <repository>
     <id>jboss-public-repository-group</id>
     <name>JBoss Public Maven Repository Group</name>

<url>https://repository.jboss.org/nexus/content/groups/public-jboss/</url>
            <layout>default</layout>
            <releases>
         <enabled>true</enabled>
         <updatePolicy>never</updatePolicy>
            </releases>
            <snapshots>
         <enabled>true</enabled>
         <updatePolicy>never</updatePolicy>
            </snapshots>
        </repository>
    </repositories>
    <pluginRepositories>
        <pluginRepository>
     <id>jboss-datagrid-repository-group</id>
     <name>JBoss Data Grid Maven Repository</name>
     <url>file:///path/to/repo/jboss-datagrid-{VERSION}-maven-
repository</url>
     <layout>default</layout>
            <releases>
             <enabled>true</enabled>
             <updatePolicy>never</updatePolicy>
            </releases>
            <snapshots>
             <enabled>false</enabled>
             <updatePolicy>never</updatePolicy>
            </snapshots>
        </pluginRepository>
        <pluginRepository>
            <id>jboss-public-repository-group</id>
     <name>JBoss Public Maven Repository Group</name>

<url>https://repository.jboss.org/nexus/content/groups/public-jboss/</url>
            <layout>default</layout>
            <releases>
         <enabled>true</enabled>
         <updatePolicy>never</updatePolicy>
            </releases>
            <snapshots>
         <enabled>true</enabled>
         <updatePolicy>never</updatePolicy>
            </snapshots>
        </pluginRepository>
    </pluginRepositories>
</profile>
```

Enable the profile by ensuring the following is included in the **`settings.xml`** file:

```
<activeProfiles>
    <!-- make the profile active by default -->
      <activeProfile>jboss-datagrid-repository</activeProfile>
</activeProfiles>
```

-

If you are using a build system that does not support declarative dependency management, add the contents of the **`client/java/`** directory, included in the Red Hat JBoss Data Grid package to the build classpath.

Report a bug

CHAPTER 7. RUN RED HAT JBOSS DATA GRID IN LIBRARY MODE (SINGLE-NODE SETUP)

7.1. CREATE A MAIN METHOD IN THE QUICKSTART CLASS

Create a new Quickstart class by following the outlined steps:

Prerequisites

These quickstarts use the Infinispan quickstarts located at https://github.com/infinispan/infinispan-quickstart. The following procedure uses the **`infinispan-quickstart/embedded-cache`** quickstart.

Procedure 7.1. Create a Main Method in the Quickstart Class

1. **Create the Quickstart.java File**
   Create a file called **`Quickstart.java`** at your project's location.

2. **Add the Quickstart Class**
   Add the following class and method:

```
package com.mycompany.app;

import org.infinispan.manager.DefaultCacheManager
import org.infinispan.Cache

public class Quickstart {
  public static void main(String args[]) throws Exception {
       Cache<Object, Object> cache = new
DefaultCacheManager().getCache();
    }
 }
```

3. **Copy Dependencies and Compile Java Classes**
   Use the following command to copy all project dependencies to a directory and compile the Java classes from our project:

```
$ mvn clean compile dependency:copy-dependencies -DstripVersion
```

4. **Run the Main Method**
   Use the following command to run the main method:

```
$ java -cp target/classes/:target/dependency/*
com.mycompany.app.Quickstart
```

Report a bug

7.2. USE THE DEFAULT CACHE

7.2.1. Add and Remove Data from the Cache

Red Hat JBoss Data Grid offers an interface that is similar to the proposed JSR-107 API to access and alter data stored in a cache.

The following procedure is an example that defines what each line entered into the DefaultCacheQuickstart.java file does:

Procedure 7.2. Add and Remove Data from the Cache

1. Add an entry, replacing *key* and *value* with the desired key and value:

```
cache.put("key", "value");
```

2. Confirm that the entry is present in the cache:

```
assertEquals(1, cache.size());
assertTrue(cache.containsKey("key"));
```

3. Remove the entry from the cache:

```
Object v = cache.remove("key");
```

4. Confirm that the entry is no longer present in the cache:

```
assertEquals("value", v);
assertTrue(cache.isEmpty());
```

Report a bug

7.2.2. Adding and Replacing a Key Value

Red Hat JBoss Data Grid offers a thread-safe data structure.

The following procedure is an example that defines what each line entered into the **`DefaultCacheQuickstart.java`** file does:

Procedure 7.3. Adding and Replacing a Key Value

- Add an entry **`key`** with **`value`** as the key's value.

```
cache.put("key", "value");
```

Procedure 7.4. Replacing a Key Value

1. The following code searches for keys (named **`key`** and **`key2`**). If the two specific keys beings searched for are not found, JBoss Data Grid creates two new keys with the specified key names and values.

```
cache.putIfAbsent("key", "newValue");
cache.putIfAbsent("key2", "value2");
```

2. The following code confirms that the value of the stored key equals the value we wanted to store.

```
assertEquals(cache.get("key"), "value");
assertEquals(cache.get("key2"), "value2");
```

See Also:

- Section A.1, “About Key-Value Pairs”

Report a bug

7.2.3. Adjust Data Life

Red Hat JBoss Data Grid entries are immortal by default, but these settings can be altered.

The following procedure is an example that defines what each line entered into the **`DefaultCacheQuickstart.java`** file does:

Procedure 7.5. Adjust the Data Life

1. Alter the key's ***`lifespan`*** value:

```
cache.put("key", "value", 5, TimeUnit.SECONDS);
```

2. Check if the cache contains the key:

```
assertTrue(cache.containsKey("key"));
```

3. After the allocated ***`lifespan`*** time has expired, the key is no longer in the cache:

```
Thread.sleep(10000);
assertFalse(cache.containsKey("key"));
```

Report a bug

7.2.4. Default Data Mortality

As a default, newly created entries do not have a life span or maximum idle time value set. Without these two values, a data entry will never expire and is therefore known as immortal data.

Report a bug

7.2.5. Register the Named Cache Using XML

To configure the named cache declaratively (using XML) rather than programmatically, configure the **`infinispan.xml`** file.

The **`infinispan.xml`** file is located in https://github.com/infinispan/infinispan-quickstart in the **`infinispan-quickstart/embedded-cache/src/main/resources`** folder.

Report a bug

CHAPTER 8. RUN RED HAT JBOSS DATA GRID IN LIBRARY MODE (MULTI-NODE SETUP)

8.1. SHARING JGROUP CHANNELS

Red Hat JBoss Data Grid offers an easy to use form of clustering using JGroups as the network transport. As a result, JGroups manages the initial operations required to form a cluster for JBoss Data Grid.

All caches created from a single CacheManager share the same JGroups channel by default. This JGroups channel is used to multiplex replication/distribution messages.

In the following example, all three caches used the same JGroups channel:

```
EmbeddedCacheManager cm = $LOCATION
Cache<Object, Object> cache1 = cm.getCache("replSyncCache");
Cache<Object, Object> cache2 = cm.getCache("replAsyncCache");
Cache<Object, Object> cache3 = cm.getCache("invalidationSyncCache");
```

Substitute *$LOCATION* with the location for the CacheManager.

Report a bug

8.2. RUN RED HAT JBOSS DATA GRID IN A CLUSTER

The clustered quickstarts for Red Hat JBoss Data Grid are based on the quickstarts found in https://github.com/infinispan/infinispan-quickstart/tree/master/clustered-cache.

Report a bug

8.2.1. Compile the Project

Use Maven to compile your project with the following command:

```
$ mvn clean compile dependency:copy-dependencies -DstripVersion
```

Report a bug

8.2.2. Run the Clustered Cache with Replication Mode

To run Red Hat JBoss Data Grid's replication mode example of a clustered cache, launch two nodes from different consoles.

Procedure 8.1. Run the Clustered Cache with Replication Mode

1. Use the following command to launch the first node:

```
$ java -cp target/classes/:target/dependency/*
org.infinispan.quickstart.clusteredcache.replication.Node0
```

2. Use the following command to launch the second node:

```
$ java -cp target/classes/:target/dependency/*
org.infinispan.quickstart.clusteredcache.replication.Node1
```

Result

JGroups and JBoss Data Grid initialized on both nodes. After approximately fifteen seconds, the cache entry log message appears on the console of the first node.

Report a bug

8.2.3. Run the Clustered Cache with Distribution Mode

To run Red Hat JBoss Data Grid's distribution mode example of a clustered cache, launch three nodes from different consoles.

Procedure 8.2. Run the Clustered Cache with Distribution Mode

1. Use the following command to launch the first node:

```
$ java -cp target/classes/:target/dependency/*
org.infinispan.quickstart.clusteredcache.distribution.Node0
```

2. Use the following command to launch the second node:

```
$ java -cp target/classes/:target/dependency/*
org.infinispan.quickstart.clusteredcache.distribution.Node1
```

3. Use the following command to launch the third node:

```
$ java -cp target/classes/:target/dependency/*
org.infinispan.quickstart.clusteredcache.distribution.Node2
```

Result

JGroups and JBoss Data Grid initialized on the three nodes. After approximately fifteen seconds, the ten entries added by the third node can be seen as they are distributed to the first and second nodes.

Report a bug

8.2.4. Configure the Cluster

8.2.4.1. Configuring the Cluster

Use the following steps to add and configure your cluster:

Procedure 8.3. Configure the Cluster

1. Add the default configuration for a new cluster
2. Customize the default cluster configuration according to the requirements of your network. This can be done declaratively (using XML) or programmatically.
3. Configure the replicated or distributed data grid.

Report a bug

8.2.4.2. Add the Default Cluster Configuration

Add a cluster configuration to ensure that Red Hat JBoss Data Grid is aware that a cluster exists and is defined. The following is a default configuration that serves this purpose:

```
new ConfigurationBuilder()
   .clustering().cacheMode(CacheMode.REPL_SYNC)
   .build()
```

NOTE

Use **`GlobalConfiguration.clusteredDefault()`** to quickly create a preconfigured and cluster-aware **`GlobalConfiguration`** for clusters. This configuration can also be customized.

Report a bug

8.2.4.3. Customize the Default Cluster Configuration

Depending on the network requirements, you may need to customize your JGroups configuration.

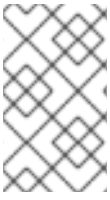
Programmatic Configuration:

Use the following GlobalConfiguration code to specify the name of the file to use for JGroups configuration:

```
new
GlobalConfigurationBuilder().transport().addProperty("configurationFile",
"jgroups.xml")
   .build()
```

Replace **`jgroups.xml`** with the desired file name.

The **`jgroups.xml`** file is located at **`$Infinispan-Quickstart/clustered-cache/src/main/resources/`**.



NOTE

To bind JGroups solely to your loopback interface (to avoid any configured firewalls), use the system property **`-Djgroups.bind_addr="127.0.0.1"`**. This is particularly useful to test a cluster where all nodes are on a single machine.

Declarative Configuration:

Use the following XML snippet in the **`infinispan.xml`** file to configure the JGroups properties to use Red Hat JBoss Data Grid's XML configuration:

```
<global>
   <transport>
      <properties>
         <property name="configurationFile" value="jgroups.xml"/>
```

```
      </properties>
   </transport>
</global>
```

Report a bug

8.2.4.4. Configure the Replicated Data Grid

Red Hat JBoss Data Grid's replicated mode ensures that every entry is replicated on every node in the data grid.

This mode offers security against data loss due to node failures and excellent data availability. These benefits are at the cost of limiting the storage capacity to the amount of storage available on the node with the least memory.

Programmatic Configuration:

Use the following code snippet to programmatically configure the cache for replication mode (either synchronous or asynchronous):

```
private static EmbeddedCacheManager createCacheManagerProgramatically() {
   return new DefaultCacheManager(
      new GlobalConfigurationBuilder()
         .transport().addProperty("configurationFile", "jgroups.xml")
         .build(),
      new ConfigurationBuilder()
         .clustering().cacheMode(CacheMode.REPL_SYNC)
         .build()
   );
}
```

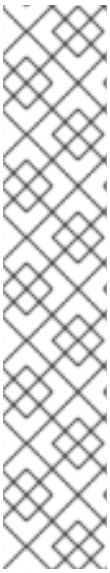
Declarative Configuration:

Edit the **`infinispan-replication.xml`** file to include the following XML code to declaratively configure the cache for replication mode (either synchronous or asynchronous):

```
<infinispan xsi:schemaLocation="urn:infinispan:config:5.1
http://www.infinispan.org/schemas/infinispan-config-5.1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:infinispan:config:5.1">
   <global>
      <transport>
         <properties>
            <property name="configurationFile" value="jgroups.xml"/>
         </properties>
      </transport>
   </global>
   <default>
      <clustering mode="replication">
         <sync/>
      </clustering>
   </default>
</infinispan>
```

Use the following code to initialize and return a DefaultCacheManager with the XML configuration file:

```
private static EmbeddedCacheManager createCacheManagerFromXml() throws
IOException {
   return new DefaultCacheManager("infinispan-replication.xml");}
```



NOTE

JBoss EAP includes its own underlying JMX. This can cause a collision when using the sample code with JBoss EAP and display an error such as **`org.infinispan.jmx.JmxDomainConflictException: Domain already registered org.infinispan`**.

To avoid this, configure global configuration as follows:

```
GlobalConfiguration glob = new GlobalConfigurationBuilder()
 .clusteredDefault()
        .globalJmxStatistics()
          .allowDuplicateDomains(true)
          .enable()
        .build();
```

Report a bug

8.2.4.5. Configure the Distributed Data Grid

Red Hat JBoss Data Grid's distributed mode ensures that each entry is stored on a subset of the total nodes in the data grid. The number of nodes in the subset is controlled by the ***`numOwners`*** parameter to indicate how many owners each entry has.

Distributed mode offers increased storage capacity but increased access times and less durability (protection against node failures). Adjust the ***`numOwners`*** value to set the desired trade off between space, durability and availability. Durability is further improved by JBoss Data Grid's topology aware consistent hash, which locates entry owners across a variety of data centers, racks and nodes.

Programmatic Configuration:

Use the following code snippet to programmatically configure the cache for distributed mode (either synchronous or asynchronous):

```
new ConfigurationBuilder()
   .clustering()
      .cacheMode(CacheMode.DIST_SYNC)
      .hash().numOwners(2)
   .build()
```

Declarative Configuration:

Edit the **`cfg.xml`** file to include the following XML code to declaratively configure the cache for distributed mode (either synchronous or asynchronous):

```
<default>
   <clustering mode="distribution">
      <sync/>
```

```
        <hash numOwners="2"/>
    </clustering>
</default>
```

Report a bug

CHAPTER 9. MONITOR RED HAT JBOSS DATA GRID APPLICATIONS IN RED HAT JBOSS EAP

Red Hat JBoss Data Grid library applications (in the form of WAR or EAR files) can be deployed within JBoss Enterprise Application Server 6 (or better) and then monitored using JBoss Operations Network.

Report a bug

9.1. PREREQUISITES

The following are prerequisites to monitor a Red Hat JBoss Data Grid library application in JBoss Enterprise Application Platform:

- Install and configure JBoss Enterprise Application Platform 6 (or better).
- Install and configure JBoss Operations Network 3.1 (or better) or RHQ (4.5.1 or better) from http://www.jboss.org/rhq.
- Install and configure JBoss Data Grid (6.1 or better) Library mode plug-in.

Report a bug

9.2. MONITOR RED HAT JBOSS DATA GRID APPLICATIONS IN RED HAT JBOSS EAP

Ensure that all requirements outlined as prerequisites are met. Follow the listed steps to monitor Red Hat JBoss Data Grid applications in JBoss Enterprise Application Platform using JBoss Operations Network or RHQ.

Procedure 9.1. Monitor JBoss Data Grid Applications in JBoss Enterprise Application Platform

1. **Configure RHQ/JBoss Operations Network**
   Add an RHQ/JBoss Operations Network specific property (named ***`org.rhq.resourceKey`***) to the **`/bin/standalone.conf`** file as follows:

   ```
   JAVA_OPTS="$JAVA_OPTS -Dorg.rhq.resourceKey=MyEAP"
   ```

   This command adds the property to the JBoss Enterprise Application Platform's command line indirectly.

2. **Check RHQ/JBoss Operations Network is Running Using a Full JDK**
   Ensure that the RHQ/JBoss Operations Network agent started using a full JDK instead of a JRE. This is because the agent requires access to the JDK's **`tools.jar`** file.

   To configure your RHQ/JBoss Operations Network agent to use the JDK, follow the instructions relevant to your operating system:

   a. For Linux users, set the ***`RHQ_AGENT_JAVA_HOME`*** environment variable to the JDK home directory in the agent's **`rhq-agent-env.sh`** file.

   b. For Windows users, set the ***`RHQ_AGENT_JAVA_HOME`*** environment variable to the JDK home directory in the agent's **`rhq-agent-env.bat`** file.

3. **Ensure the Agent is Local to the JBoss Enterprise Application Platform Instance**
   Ensure that the RHQ/JBoss Operations Network agent runs locally to and under the same user as the JBoss Application Platform instance. This is required for the Java Attach API to connect to the process.

4. **Import Resources to the Agent Inventory**
   RHQ/JBoss Operations Network can now discover resources. These resources can subsequently be imported into the agent inventory.

   When a JBoss Data Grid user deployment enables JMX statistics to expose JBoss Data Grid Cache Managers or caches, the resources appear as children resources of the JBoss Enterprise Application Platform instance.

Report a bug

PART V. RED HAT JBOSS DATA GRID QUICKSTARTS

CHAPTER 10. THE HELLO WORLD QUICKSTART

Hello World is a simple quickstart that illustrates how to store data to and retrieve data from a cache using Red Hat JBoss Data Grid. For this quickstart, users can access the cache in two ways:

- from a servlet.
- from a JSF page using request scoped beans.

All libraries (JAR files) bundles with the application are deployed to
JBoss Enterprise Application Platform 6.x or WildFly 7.x. JBoss Data Grid's Library mode only allows local access to a single node in a distributed cluster. This mode also allows the application to access the data grid functionality within a virtual machine in the target container.

IMPORTANT

The Hello World quickstart works only in JBoss Data Grid's Library mode.

Location

JBoss Data Grid's Hello World quickstart is available at the following location: **`jboss-datagrid-{VERSION}-quickstarts/`**

Report a bug

10.1. QUICKSTART PREREQUISITES

The prerequisites for this quickstart are as follows:

- Java 6.0 (Java SDK 1.6) or better
- JBoss Enterprise Application Platform 6.x or WildFly 7.x
- Maven 3.0 or better
- Configure the Maven Repository. For details, see Chapter 3, *Install and Use the Maven Repositories*

Report a bug

10.2. START TWO APPLICATION SERVER INSTANCES

Before deploying the Hello World quickstart, start two instances of your application server (either JBoss Enterprise Application Platform 6.x or WildFly 7.x). Start two instances of the selected application server as outlines in the following procedures.

Procedure 10.1. Start the First Application Server Instance

1. **Navigate to the Root Directory**
   In the command line terminal, navigate to the root for your JBoss server directory.
2. **Start the First Application Server**
   Depending on your operating system, use the appropriate command from the following to start the first instance of your selected application server:

a. For Linux:

```
$JBOSS_HOME/bin/standalone.sh
```

b. For Windows:

```
$JBOSS_HOME\bin\standalone.bat
```

Procedure 10.2. Start the Second Application Server Instance

1. **Clone the Application Server**
   Create a copy of the selected JBoss Server to create a second instance.

2. **Navigate to the Root Directory**
   In the command line terminal, navigate to the root for your JBoss server directory.

3. **Start the Second Application Server**
   Use the appropriate command for your operating system from the following commands. This command starts the server with the provided port offset to ensure that both the server instances run on the same host.

   a. For Linux:

   ```
   $JBOSS_HOME2/bin/standalone.sh -Djboss.socket.binding.port-
   offset=100
   ```

   b. For Windows:

   ```
   $JBOSS_HOME2\bin\standalone.bat -Djboss.socket.binding.port-
   offset=100
   ```

Report a bug

10.3. BUILD AND DEPLOY THE HELLO WORLD QUICKSTART

IMPORTANT

Before building and deploying the quickstart, ensure that all the listed prerequisites are met and that the two application server instances are running.

Procedure 10.3. Build and Deploy the Hello World Quickstart

1. **Navigate to the Required Directory**
   In the command line terminal, navigate to the root directory of the quickstart on the command line interface.

2. **Build and Deploy to the First Application Server Instance**
   Use the following command to build and deploy the quickstart to the first application server instance as follows:

   ```
   # mvn clean package jboss-as:deploy
   ```

This command deploys **target/jboss-helloworld-jdg.war** to the first running server instance.

3. **Build and Deploy to the Second Application Server Instance**
   Use the following command to build and deploy the quickstart to the second application server instance with the specified ports as follows:

   ```
   # mvn clean package jboss-as:deploy -Djboss-as.port=10099
   ```

   This command deploys **target/jboss-helloworld-jdg.war** to the second running server instance.

Report a bug

10.4. ACCESS THE RUNNING APPLICATION

The Hello World quickstart application runs on the following URLs:

- First Server Instance: http://localhost:8080/jboss-helloworld-jdg
- Second Server Instance: http://localhost:8180/jboss-helloworld-jdg

Report a bug

10.5. TEST REPLICATION ON THE APPLICATION

The following is a simple procedure outlining cache entries being replicated from the first server instance to the second instance.

Procedure 10.4. Test Replication on the Application

1. **Access the First Server**
   Access the first application server and enter the key and value.

   a. Access the first application server in a browser window using the following URL:

      ```
      http://localhost:8080/jboss-helloworld-jdg
      ```

   b. Insert the key **foo**.

   c. Insert the value **bar**.

2. **Access the Second Server**
   Access the second application server and enter the key and value.

   a. Access the second application server in a browser window using the following URL:

      ```
      http://localhost:8180/jboss-helloworld-jdg
      ```

   b. Click **Get Some**.

   c. Get the key **foo**.

d. Click **Put Some More**.

e. Insert the key **`mykey`**.

f. Insert the value **`myvalue`**.

3. **Get All Keys and Values**
Access the first server and request all keys.

a. Access the first application server in a browser window using the following URL:

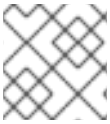
```
http://localhost:8080/jboss-helloworld-jdg
```

b. Click **Get Some**.

c. Click **Get All** to request all key and values.

Result

As the results of the last step show, all the data added at each server has been replicated to the other server.



NOTE

Entries expire after **`60`** seconds from the most recent update.

Directly Access Keys in the Cache

To interact with predefined servlets or to directly store and retrieve keys from the cache, use the following URLs:

```
http://localhost:8080/jboss-helloworld-jdg/TestServletPut
```

```
http://localhost:8180/jboss-helloworld-jdg/TestServletGet
```

Report a bug

10.6. REMOVE THE APPLICATION

Use the following procedure to remove the Hello World application:

Procedure 10.5. Remove the Application

1. **Start the Application Servers**
Ensure that both server instances are running.

2. **Navigate to the Root**
In the command line terminal, navigate to the root directory of the quickstart

3. **Remove the Archive**
Use the following commands to remove the archive from both the server instances.

a. Remove the archive from the first server as follows:

```
mvn jboss-as:undeploy
```

b. Remove the archive from the second server as follows:

```
mvn jboss-as:undeploy -Djboss-as.port=10099
```

Report a bug

10.7. DEBUG THE APPLICATION

To debug the source code for this application, run either one of the following commands to pull them down to your local repository:

```
# mvn dependency:source
```

```
# mvn dependency:resolve -Dclassifier=javadoc
```

IMPORTANT

If debugged or altered, the quickstarts are no longer supported by Red Hat.

Report a bug

CHAPTER 11. THE CARMART QUICKSTARTS

11.1. ABOUT THE CARMART QUICKSTART

Red Hat JBoss Data Grid includes a transactional and non-transactional CarMart quickstart. The CarMart quickstart is a simple web application that uses JBoss Data Grid instead of a relational database. Information about each car is stored in a cache. Caches are configured declaratively or programmatically depending on the usage mode.

Features

The CarMart quickstart offers the following features:

- List all cars
- Add new cars
- Remove cars
- View statistics for caches, such as hits, stores, and retrievals

Usage Modes

The CarMart quickstart can be used in the following JBoss Data Grid usage modes:

- Remote Client-Server Mode, where the application includes the Hot Rod client to communicate with a remote JBoss Data Grid server.
- Library Mode, where all libraries are bundled with the application in the form of `jar` files.

Location

JBoss Data Grid's CarMart quickstart is available at the following location: `jboss-datagrid-{VERSION}-quickstarts/`

IMPORTANT

The Carmart Quickstarts can only be deployed in JBoss Enterprise Application Platform, not in JBoss Data Grid as the latter is not supported.

Report a bug

11.2. ABOUT THE CARMART TRANSACTIONAL QUICKSTART

The transactional version of the CarMart quickstart is a simple web application that uses Red Hat JBoss Data Grid instead of a relational database. Information about each car is stored in a cache. Caches are configured declaratively or programmatically (depending on the usage mode) and run in the same Java Virtual Machine (JVM) as the web application.

Features

The Transactional CarMart Quickstart offers the following features:

- List all cars
- Add new cars

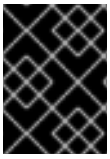
- Add new cars with rollback
- Remove cars
- View statistics for caches, such as hits, stores, and retrievals

Usage Modes

The Transactional CarMart Quickstart can only be used in JBoss Data Grid's Library mode. A standalone transaction manager from JBoss Transactions is used when the Transactional CarMart Quickstart is run in Red Hat JBoss Web Server 2.

Location

JBoss Data Grid's Transactional CarMart Quickstart can be found at the following location: **`jboss-datagrid-{VERSION}-quickstarts/carmart-tx`**



IMPORTANT

The Carmart Quickstarts can only be deployed in JBoss Enterprise Application Platform, not in JBoss Data Grid as the latter is not supported.

Report a bug

11.3. DIFFERENCES BETWEEN THE CARMART AND TRANSACTIONAL QUICKSTARTS

Despite the similarity in steps to build, deploy and remove the transactional and non-transactional CarMart quickstarts, some differences must be noted. The following is a list of such differences:

- CarMart is available for both Remote Client-Server Mode and Library Mode. Transactional CarMart is only available in Library Mode because transactions are not available in Remote Client-Server Mode.
- The Transactional Quickstart also displays how a transaction rollback occurs. Use the **Add car with rollback** button to view the rollback. The CarMart example has a simple **Add car** button instead.

Report a bug

11.4. QUICKSTART PREREQUISITES

The prerequisites for this quickstart are as follows:

- Java 6.0 (Java SDK 1.6) or better
- JBoss Enterprise Application Platform 6.x or WildFly 7.x
- Maven 3.0 or better
- Configure the Maven Repository. For details, see Chapter 3, *Install and Use the Maven Repositories*

Report a bug

11.5. THE CARMART QUICKSTART USING WILDFLY

11.5.1. Build and Deploy the CarMart Quickstart to WildFly

The following procedure provides directions to build and deploy the CarMart application to WildFly.

NOTE

The directions for this quickstart are for use with the selected version of WildFly, not Red Hat JBoss Data Grid itself. JBoss Data Grid does not support application deployment as described in this quickstart.

Prerequisites

Prerequisites for this procedure are as follows:

1. Obtain the supported JBoss Data Grid Library Mode distribution files.
2. Ensure that the JBoss Data Grid and JBoss Enterprise Application Platform Maven repositories are installed and configured. For details, see Chapter 3, *Install and Use the Maven Repositories*
3. Select a JBoss server to use (JBoss Enterprise Application Platform 6 (or better) or WildFly 7 (or better).

Procedure 11.1. Build and Deploy CarMart to WildFly

1. **Start WildFly**
   Navigate to the root of the WildFly server directory in a terminal window and enter the following command:

   For Linux users:

   ```
   $JBOSS_HOME/bin/standalone.sh
   ```

   For Windows users:

   ```
   $JBOSS_HOME\bin\standalone.bat
   ```

2. **Build and Deploy the Application**
   Use the following command to build and deploy the application using Maven:

   ```
   $ mvn clean package jboss-as:deploy
   ```

Result

The target war file (**`target/jboss-carmart.war`**) is deployed to the running instance of WildFly.

Report a bug

11.5.2. View the CarMart Quickstart on WildFly

The following procedure outlines how to view the CarMart quickstart on WildFly:

Prerequisite

The CarMart quickstart must be built and deployed to be viewed.

Procedure 11.2. View the CarMart Quickstart on WildFly

- To view the application, use your browser to navigate to the links provided.

  When using the non transactional Carmart Quickstart, use the following link:

  ```
  http://localhost:8080/jboss-carmart
  ```

  When using the transactional Carmart Quickstart, use the following link:

  ```
  http://localhost:8080/jboss-carmart-tx
  ```

Report a bug

11.5.3. Remove the CarMart Quickstart from WildFly

The following procedure provides directions to remove a deployed application from WildFly.

Procedure 11.3. Remove an Application from the WildFly

- To remove an application, use the following command:

  ```
  $ mvn jboss-as:undeploy
  ```

Report a bug

11.6. THE CARMART QUICKSTART TO A SERVER

11.6.1. Build and Deploy the CarMart Quickstart to the Server

The following procedure provides directions to build and deploy the CarMart quickstart to the server.

Prerequisites

Prerequisites for this procedure are as follows:

1.
2. Ensure that the JBoss Data Grid and JBoss Enterprise Application Platform Maven repositories are installed and configured. For details, see Chapter 3, *Install and Use the Maven Repositories*
3. Select JBoss Web Server 2 (or better) for your application and install it.

Procedure 11.4. Build the CarMart Quickstart to the Server (Library Mode)

1. **Start the Server**
   Run the selected server by navigating to the root directory in a terminal window and enter the following command:

   For Linux users:

```
$JBOSS_EWS_HOME/tomcat7/bin/catalina.sh run
```

For Windows users:

```
$JBOSS_EWS_HOME\tomcat7\bin\catalina.bat run
```

2. **Build and Deploy your Application**
   Use the following command to build and deploy your application using Maven:

```
$ mvn -Plibrary-tomcat clean package tomcat:deploy
```

Result

The target war file (**`target/jboss-carmart.war`**) is deployed to the running instance of the selected server.

Report a bug

11.6.2. View the CarMart Quickstart on the Server

The following procedure outlines how to view the CarMart quickstart on the server:

Prerequisite

The CarMart quickstart must be built and deployed to be viewed.

Procedure 11.5. View the CarMart Quickstart

- To view the application, use your browser to navigate to the following link:

```
http://localhost:8080/jboss-carmart
```

Report a bug

11.6.3. Remove the CarMart Quickstart from the Server

The following procedure provides directions to remove an already deployed application from the server.

Procedure 11.6. Remove an Application from the Server

- To remove an application, use the following command:

```
$ mvn tomcat:undeploy -Plibrary-tomcat
```

Report a bug

11.7. THE CARMART QUICKSTART IN REMOTE CLIENT-SERVER MODE

11.7.1. Build and Deploy the CarMart Quickstart in Remote Client-Server Mode

This quickstart accesses Red Hat JBoss Data Grid via Hot Rod. This feature is not available for the Transactional CarMart quickstart.

IMPORTANT

This quickstart deploys to JBoss Enterprise Application Platform or WildFly. The application cannot be deployed to JBoss Data Grid because it does not support application deployment.

Prerequisites

Prerequisites for this procedure are as follows:

1. Obtain the most recent supported JBoss Data Grid Remote Client-Server Mode distribution files from Red Hat.
2. Ensure that the JBoss Data Grid and JBoss Enterprise Application Platform Maven repositories are installed and configured. For details, see Chapter 3, *Install and Use the Maven Repositories*
3. Select a JBoss server to use (JBoss Enterprise Application Platform 6 (or better) or Wildfly 7 (or better). Navigate to the root of the JBoss server directory in a terminal window and enter the following command:

   For Linux users:

   ```
   $JBOSS_HOME/bin/standalone.sh
   ```

   For Windows users:

   ```
   $JBOSS_HOME\bin\standalone.bat
   ```

Procedure 11.7. Build and Deploy the CarMart Quickstart in Remote Client-Server Mode

1. **Configure the Standalone File**
   Add the following configuration to the **`standalone.xml`** file located in the **`$JDG_HOME/standalone/configuration/`** directory.

   a. Add the following configuration within the infinispan subsystem tags:

   ```
   <local-cache name="carcache"
         start="EAGER"
         batching="false"
         statistics="true">
    <eviction strategy="LIRS"
       max-entries="4"/>
   </local-cache>
   ```

   NOTE

   If the ***`carcache`*** element already exists in your configuration, replace it with the provided configuration.

2. **Start the JBoss Data Grid Server**

Run the following script to start the JBoss Data Grid Server:

```
$JDG_HOME/bin/standalone.sh -Djboss.socket.binding.port-offset=100
```

3. **Start the JBoss Server**
   Run the following script to start the JBoss server instance where your application will deploy:

```
$JBOSS_HOME/bin/standalone.sh
```

4. **Optional: Specify the Host and Port Address**
   The application uses the values in the **`jboss-datagrid-{VERSION}-quickstarts/carmart/src/main/resources/META-INF/datagrid.properties`** file to locate the JBoss Data Grid server. If your JBoss Data Grid server is not running using the default host and port values, edit the file and insert the correct host and port values, as follows:

```
datagrid.host=localhost
datagrid.hotrod.port=11322
```

5. **Build and Deploy the Application**
   Use the following command to build and deploy your application in the relevant directory:

```
$ mvn clean package -Premote
```

Report a bug

11.7.2. View the CarMart Quickstart in Remote Client-Server Mode

The following procedure outlines how to view the CarMart quickstart in Red Hat JBoss Data Grid's Remote Client-Server Mode:

Prerequisite

The CarMart quickstart must be built and deployed be viewed.

Procedure 11.8. View the CarMart Quickstart in Remote Client-Server Mode

- Visit the following link in a browser window to view the application:

```
http://localhost:8080/jboss-carmart
```

Report a bug

11.7.3. Remove the CarMart Quickstart in Remote Client-Server Mode

The following procedure provides directions to remove an already deployed application in Red Hat JBoss Data Grid's Remote Client-Server mode.

Procedure 11.9. Remove an Application in Remote Client-Server Mode

- To remove an application, use the following command:

```
$ mvn jboss-as:undeploy -Premote
```

Report a bug

CHAPTER 12. THE FOOTBALL QUICKSTART ENDPOINT EXAMPLES

The Football application is a simple example to illustrate the use of Red Hat JBoss Data Grid endpoints, namely Hot Rod, REST, and Memcached. Each example shows one of these protocols used to connect to JBoss Data Grid to remotely store, retrieve, and remove data from caches.

Each application is a variation of a simple football team manager utility as a console application.

Features

The following features are available with the example Football Manager application:

- Add a team
- Add players
- Remove all entities (teams and players)
- Listing all teams and players

Location

JBoss Data Grid's Football quickstart can be found at the following locations:

- **`jboss-datagrid-{VERSION}-quickstarts/rest-endpoint`**
- **`jboss-datagrid-{VERSION}-quickstarts/hotrod-endpoint`**
- **`jboss-datagrid-{VERSION}-quickstarts/memcached-endpoint`**

Report a bug

12.1. QUICKSTART PREREQUISITES

The prerequisites for this quickstart are as follows:

- Java 6.0 (Java SDK 1.6) or better
- JBoss Enterprise Application Platform 6.x or WildFly 7.x
- Maven 3.0 or better
- Configure the Maven Repository. For details, see Chapter 3, *Install and Use the Maven Repositories*

Report a bug

12.2. BUILD THE FOOTBALL APPLICATION

The following procedure outlines the steps to build a football manager application as an example of REST, Hot Rod and memcached endpoints in Red Hat JBoss Data Grid.

IMPORTANT

JBoss Data Grid does not support deploying applications, therefore this quickstart cannot be installed as a deployment.

Prerequisites

Prerequisites for this procedure are as follows:

1. Obtain the most recent supported JBoss Data Grid Remote Client-Server Mode distribution files from Red Hat.
2. Ensure that the JBoss Data Grid and JBoss Enterprise Application Platform Maven repositories are installed and configured. For details, see Chapter 3, *Install and Use the Maven Repositories*

Procedure 12.1. Build the Football Application

1. **Add Configurations**
   Edit the **`standalone.xml`** file (located at **`$JDG_HOME/standalone/configuration/`**) to add definitions for the datasource and infinispan subsystems.

   a. Add the following subsystem definition for the datasource:

```
<subsystem xmlns="urn:jboss:domain:datasources:1.0">

<!-- Define this Datasource with jndi
name java:jboss/datasources/ExampleDS -->

    <datasources>
        <datasource jndi-name="java:jboss/datasources/ExampleDS"
      pool-name="ExampleDS"
      enabled="true"
      use-java-context="true">

            <!-- The connection URL uses H2 Database
  Engine with in-memory database called test -->

            <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-
1</connection-url>

            <!-- JDBC driver name -->
            <driver>h2</driver>

            <!-- Credentials -->
            <security>
                <user-name>sa</user-name>
                <password>sa</password>
            </security>
        </datasource>

        <!-- Define the JDBC driver called 'h2' -->
        <drivers>
            <driver name="h2" module="com.h2database.h2">
                <xa-datasource-
class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
            </driver>
```

```
        </drivers>

    </datasources>
</subsystem>
```

b. Add the following subsystem definition for infinispan:

```
<subsystem xmlns="urn:infinispan:server:core:6.0"
      default-cache-container="local">

 <cache-container name="local"
    default-cache="default"
    statistics="true">

         <local-cache name="default"
        start="EAGER"
        statistics="true">

   <locking isolation="NONE"
     acquire-timeout="30000"
     concurrency-level="1000"
     striping="false"/>

   <transaction mode="NONE"/>

  </local-cache>

  <local-cache name="memcachedCache"
        start="EAGER"
        statistics="true">

   <locking isolation="NONE"
     acquire-timeout="30000"
     concurrency-level="1000"
     striping="false"/>
   <transaction mode="NONE"/>

  </local-cache>

         <local-cache name="namedCache"
        start="EAGER"
        statistics="true"/>

         <!-- ADD a local cache called 'teams' -->

  <local-cache name="teams"
               start="EAGER"
                batching="false"
                statistics="true">

            <!-- Disable transactions for this cache -->
   <transaction mode="NONE" />

            <!-- Define the JdbcBinaryStores
   to point to the ExampleDS previously
   defined -->
```

```
    <string-keyed-jdbc-store
     datasource="java:jboss/datasources/ExampleDS"
      passivation="false"
     preload="false"
     purge="false">

                    <!-- Define the database dialect -->
                    <property name="databaseType">H2</property>

                   <!-- specifies information about
         database table/column names
         and data types -->

     <string-keyed-table prefix="JDG">
                       <id-column name="id"
                    type="VARCHAR"/>
        <data-column name="datum"
                     type="BINARY"/>
        <timestamp-column name="version"
                          type="BIGINT"/>
     </string-keyed-table>

    </string-keyed-jdbc-store>

          </local-cache>

          <!-- End of local cache called 'teams' definition -->

  </cache-container>
</subsystem>
```

NOTE

The Hot Rod and REST endpoints use the cache named **`teams`** and memcached endpoint uses **`memcachedCache`** as a default.

2. **Disable REST Security**
   As a default, the **`standalone.xml`** configuration file protects the REST endpoint with **`BASIC`** authentication. This quickstart cannot perform authentication, therefore the REST authentication must be disabled in the REST connector by removing the ***`security-domain`*** and ***`auth-method`*** parameters. The resulting configuration (with REST authentication disabled) is as follows:

   ```
   <rest-connector virtual-server="default-host"
                   cache-container="local" />
   ```

   For more details about security, see the REST Authentication Chapter in JBoss Data Grid's *Developer Guide*.

3. **Edit the Submodule Configuration File**
   Each submodule (specifically **`hotrod-endpoint`**, **`rest-endpoint`** and **`memcached-endpoint`**) contains a configuration file (located at **`$JDG_QUICKSTART/src/main/resources/jdg.properties`**). Modify the default values in

the configuration file to set the values required for your specific JBoss Data Grid installation.

4. **Build the Application**
   Use the following command to build the example application in its directory:

   ```
   mvn clean package
   ```

   This step results in the use of Maven's shade plugin, which bundles all dependencies into a single jar file for ease of use. This file is named **`{PROTOCOL}-endpoint-quickstart.jar`**, for example **`rest-endpoint-quickstart.jar`** for the REST version.

5. **Start JBoss Data Grid**
   Run the following script to run JBoss Data Grid:

   ```
   $JDG_HOME/bin/standalone.sh
   ```

6. **Run the Application**
   Run the example application in its directory with the following command:

   ```
   mvn exec:java
   ```

Report a bug

PART VI. UNINSTALL RED HAT JBOSS DATA GRID

CHAPTER 13. REMOVE RED HAT JBOSS DATA GRID

13.1. REMOVE RED HAT JBOSS DATA GRID FROM YOUR LINUX SYSTEM

The following procedures contain instructions to remove Red Hat JBoss Data Grid from your Linux system.

WARNING

Once deleted, all JBoss Data Grid configuration and settings are permanently lost.

Procedure 13.1. Remove JBoss Data Grid from Your Linux System

1. **Shut Down Server**
   Ensure that the JBoss Data Grid server is shut down.
2. **Navigate to the JBoss Data Grid Home Directory**
   Use the command line to change into the level above the **$JDG_HOME** folder.
3. **Delete the JBoss Data Grid Home Directory**
   Enter the following command in the terminal to remove JBoss Data Grid, replacing **$JDG_HOME** with the name of your JBoss Data Grid home directory:

   ```
   $ rm -Rf $JDG_HOME
   ```

Report a bug

13.2. REMOVE RED HAT JBOSS DATA GRID FROM YOUR WINDOWS SYSTEM

The following procedures contain instructions to remove Red Hat JBoss Data Grid from your Microsoft Windows system.

WARNING

Once deleted, all JBoss Data Grid configuration and settings are permanently lost.

Procedure 13.2. Remove JBoss Data Grid from Your Windows System

1. **Shut Down Server**
   Ensure that the JBoss Data Grid server is shut down.

2. **Navigate to the JBoss Data Grid Home Directory**
   Use the Windows Explorer to navigate to the directory in which the **$JDG_HOME** folder is located.

3. **Delete the JBoss Data Grid Home Directory**
   Select the **$JDG_HOME** folder and delete it.

Report a bug

APPENDIX A. REFERENCES

A.1. ABOUT KEY-VALUE PAIRS

A key-value pair (KVP) is a set of data consisting of a key and a value.

- A key is unique to a particular data entry and is composed from data attributes of the particular entry it relates to.
- A value is the data assigned to and identified by the key.

Report a bug

APPENDIX B. MAVEN CONFIGURATION INFORMATION

B.1. INSTALL THE JBOSS ENTERPRISE APPLICATION PLATFORM REPOSITORY USING NEXUS

This example will cover the steps to install the JBoss Enterprise Application Platform 6 Maven Repository using Sonatype Nexus Maven Repository Manager. For more complete instructions, see Sonatype Nexus: Manage Artifacts.

Procedure B.1. Download the JBoss Enterprise Application Platform 6 Maven Repository ZIP archive

1. Open a web browser and access this URL: https://access.redhat.com/jbossnetwork/restricted/listSoftware.html?product=appplatform.
2. Find "Application Platform 6 Maven Repository" in the list.
3. Click the **`Download`** button to download a **`.zip`** file containing the repository.
4. Unzip the files into a directory of your choosing.

Procedure B.2. Add the JBoss Enterprise Application Platform 6 Maven Repository using Nexus Maven Repository Manager

1. Log into Nexus as an Administrator.
2. Select the **`Repositories`** section from the **Views** → **Repositories** menu to the left of your repository manager.
3. Click the **`Add...`** drop-down menu, then select **`Hosted Repository`**.
4. Give the new repository a name and ID.
5. Enter the path on disk to the unzipped repository in the field **`Override Local Storage`** Location.
6. Continue if the artifact must be available in a repository group. If not, do not continue with this procedure.
7. Select the repository group.
8. Click on the **`Configure`** tab.
9. Drag the new JBoss Maven repository from the **`Available Repositories`** list to the **`Ordered Group Repositories`** list on the left.

NOTE

The order of this list determines the priority for searching Maven artifacts.

Result

The repository is configured using Nexus Maven Repository Manager.

Report a bug

B.2. MAVEN REPOSITORY CONFIGURATION EXAMPLE

A sample Maven repository file named **example-settings.xml** is available in the root directory of the Maven repository folder after it is unzipped. The following is an excerpt that contains the relevant parts of the **example-settings.xml** file:

```
<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">

<profiles>
 <profile>
  <id>jboss-datagrid-repository</id>
  <repositories>
   <repository>
    <id>techpreview-all-repository</id>
    <name>Red Hat Enterprise Maven Repository</name>
    <url>http://maven.repository.redhat.com/techpreview/all/</url>
    <layout>default</layout>
    <releases>
     <enabled>true</enabled>
     <updatePolicy>never</updatePolicy>
    </releases>
    <snapshots>
     <enabled>false</enabled>
     <updatePolicy>never</updatePolicy>
    </snapshots>
   </repository>
   <repository>
    <id>jboss-datagrid-repository</id>
    <name>JBoss Data Grid Maven Repository</name>
    <url>file:///path/to/repo/jboss-datagrid-maven-repository</url>
    <layout>default</layout>
    <releases>
     <enabled>true</enabled>
     <updatePolicy>never</updatePolicy>
    </releases>
    <snapshots>
     <enabled>false</enabled>
     <updatePolicy>never</updatePolicy>
    </snapshots>
   </repository>
  </repositories>
  <pluginRepositories>
   <pluginRepository>
    <id>techpreview-all-repository</id>
    <name>Red Hat Enterprise Maven Repository</name>
    <url>http://maven.repository.redhat.com/techpreview/all/</url>
    <layout>default</layout>
    <releases>
```

```
      <enabled>true</enabled>
      <updatePolicy>never</updatePolicy>
     </releases>
     <snapshots>
      <enabled>false</enabled>
      <updatePolicy>never</updatePolicy>
     </snapshots>
    </pluginRepository>
    <pluginRepository>
     <id>jboss-datagrid-repository</id>
     <name>JBoss Data Grid Maven Repository</name>
     <url>file:///path/to/repo/jboss-datagrid-maven-repository</url>
     <layout>default</layout>
     <releases>
      <enabled>true</enabled>
      <updatePolicy>never</updatePolicy>
     </releases>
     <snapshots>
      <enabled>false</enabled>
      <updatePolicy>never</updatePolicy>
     </snapshots>
    </pluginRepository>
   </pluginRepositories>
  </profile>
 </profiles>
 <activeProfiles>
  <!-- make the profile active by default -->
  <activeProfile>jboss-datagrid-repository</activeProfile>
 </activeProfiles>

 </settings>
```

Report a bug

APPENDIX C. REVISION HISTORY

Revision 6.2.1-2 **Tue Mar 18 2014** **Rakesh Ghatvisave**

BZ-1076462: Standardized hyphenation in JDG modes.

Revision 6.2.1-1 **Tue Mar 11 2014** **Mandar Joshi**

BZ-1059542: Rewrote a paragraph for clarity.

BZ-1059949: Edited JVM definition.

BZ-1075622: Edited content as part of the doc review.

BZ-1074839: Added non-breaking spaces for product names.