Red Hat JBoss Data Grid 6

Getting Started Guide

An introductory guide for Red Hat JBoss Data Grid 6.

Edition 1

Last Updated: 2017-11-21

Red Hat JBoss Data Grid 6 Getting Started Guide

An introductory guide for Red Hat JBoss Data Grid 6.
Edition 1

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

David Le Sage
Red Hat Engineering Content Services
dlesage@redhat.com

B Long
Red Hat Engineering Content Services
belong@redhat.com

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

Misty Stanley-Jones
misty@redhat.com

Legal Notice

Abstract

This guide outlines introductory concepts and operations within Red Hat JBoss Data Grid 6.

Table of Contents

PREFACE

CHAPTER 1. INTRODUCING JBOSS DATA GRID 6

1.1. ABOUT JBOSS DATA GRID

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

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

Report a bug

1.2. JBOSS DATA GRID USAGE MODES

JBoss Data Grid offers two usage modes:

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.

Library mode

Library mode provides all the binaries required to build and deploy a custom runtime environment. The library usage mode allows local access to a single node in a distributed cluster. This usage mode gives the application access to data grid functionality within a virtual machine in the container being used. Supported containers include Tomcat 7 and JBoss Enterprise Application Platform 6.

Report a bug

1.3. JBOSS DATA GRID BENEFITS

Benefits of JBoss Data Grid

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 completely removes the bottleneck that exists in the vast majority of current enterprise applications.

> **Example 1.1. Massive Heap and High Availability Example**
>
> In a sample grid with one hundred 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) the resulting

> memory backed virtual heap contains 100 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

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 seed the cache and to ensure that the relevant data remains safe from corruption. The cache store also overflows data to the disk when required if a process runs 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.

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.

Report a bug

1.4. JBOSS DATA GRID VERSION INFORMATION

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 has 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.

Report a bug

1.5. JBOSS DATA GRID CACHE ARCHITECTURE

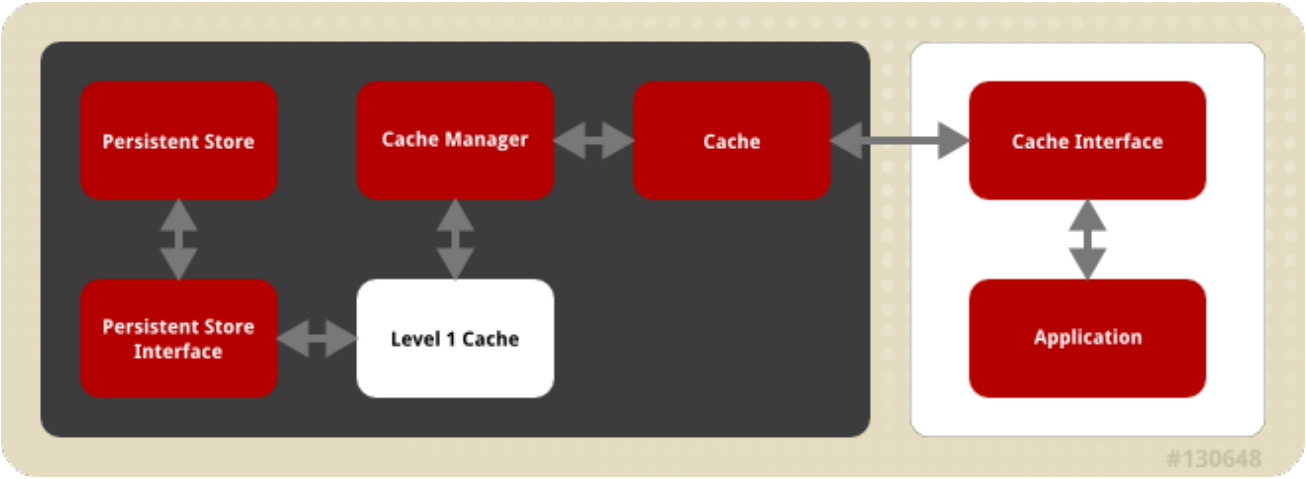


Figure 1.1. JBoss Data Grid Cache Architecture

JBoss Data Grid's cache infrastructure depicts the individual elements and their interaction with each other. For user understanding, the cache architecture diagram is separated into two parts:

- Elements that a user cannot directly interact with (depicted within a dark box), which includes the Cache, Cache Manager, Level 1 Cache, Persistent Store Interfaces and the Persistent Store.
- Elements that a user can interact directly with (depicted within a white box), which includes Cache Interfaces and the Application.

Cache Architecture Elements

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

1. The Persistent Store permanently houses cache instances and entries.
2. JBoss Data Grid offers two Persistent Store Interfaces to access the persistent store. Persistent store interfaces can be either:
    - A cache loader is a read only interface that provides a connection to a persistent data store. A cache loader can locate and retrieve data from cache instances and from the persistent store.
    - A cache store extends the cache loader functionality to include write capabilities by exposing methods that allow the cache loader to load and store states.
3. 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.
4. The Cache Manager is the primary mechanism used to retrieve a Cache instance in JBoss Data Grid, and can be used as a starting point for using the Cache.
5. The Cache houses cache instances retrieved by a Cache Manager.
6. Cache Interfaces use protocols such as Memcached and Hot Rod, or REST to interface with the cache. For details about the remote interfaces, refer to the *Developer Guide*.

- Memcached is a distributed memory object caching system used to store key-values in-memory. The Memcached caching system defines a text based, client-server caching protocol called the Memcached protocol.
- Hot Rod is a binary TCP client-server protocol used in JBoss Data Grid. It was created to overcome deficiencies in other client/server protocols, such as Memcached. Hot Rod enables clients to do smart routing of requests in partitioned or distributed JBoss Data Grid server clusters.
- The REST protocol eliminates the need for tightly coupled client libraries and bindings. The REST API introduces an overhead, and requires a REST client or custom code to understand and create REST calls.

7. An application allows the user to interact with the cache via a cache interface. Browsers are a common example of such end-user applications.

Report a bug

1.6. JBOSS DATA GRID APIS

JBoss Data Grid provides the following APIs:

- Cache
- Batching
- Grouping
- CacheStore
- Externalizable

In JBoss Data Grid's Remote Client-Server mode, only the following APIs can be used to interact with the data grid:

- 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

CHAPTER 2. PREREQUISITES FOR JBOSS DATA GRID

2.1. JBOSS DATA GRID PREREQUISITES

JBoss Data Grid requires only a Java 6.0 and above compatible Java Virtual Machine (JVM) to run. An application server is not a requirement for JBoss Data Grid.

Report a bug

2.2. JAVA VIRTUAL MACHINES (JVM)

2.2.1. Java Virtual Machine

A Java Virtual Machine (JVM) is a piece of software that is capable of running Java bytecode. The JVM creates a standard environment in which the intermediate bytecode is run. By creating the standard environment irrespective of the underlying hardware and operating system combination, it allows programmers to write their Java code once and have confidence that it can be run on any system. Red Hat recommends customers use OpenJDK as it is an open source, supported Java Virtual Machine that runs well on Red Hat Enterprise Linux systems. Windows users should install Oracle JDK 1.6.

Report a bug

2.2.2. 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 root and run the alternatives command:

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

   b. Select **`/usr/lib/jvm/jre-1.6.0-openjdk.x86_64/bin/java`**.

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

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

   d. Select **`/usr/lib/jvm/java-1.6.0-openjdk/bin/java`**.

Report a bug

CHAPTER 3. DOWNLOAD AND INSTALL JBOSS DATA GRID

3.1. DOWNLOAD JBOSS DATA GRID

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

Procedure 3.1. 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 6.0.0** for JBoss Data Grid with the Remote Client-Server usage mode or **JBoss Data Grid Library 6.0.0** for JBoss Data Grid with the Library usage mode.

Report a bug

3.2. ABOUT THE RED HAT CUSTOMER PORTAL

The *Red Hat Customer Portal* is a website where Red Hat customers download officially-supported software, manage entitlements and support contracts, contact Global Support services, and file bugs against Red Hat products. The web address to access the Customer Portal is https://access.redhat.com.

Report a bug

3.3. 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

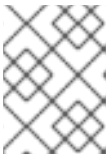
3.4. VERIFY THE DOWNLOADED FILE

Procedure 3.2. Verify the Downloaded File

1. To verify that a file downloaded from the Red Hat Customer Portal is error-free, whilst still on the portal site, go to that package's **`Software Details`** page. Here you will find **`MD5`** and **`SHA256`**"checksum" values that you will use to check the integrity of the file.

2. Open a terminal window and run either either the **md5sum** or **sha256sum** command, supplying the filename of the downloaded **ZIP** as an argument. The program will output the checksum value for the file.

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. Users of that operating system will 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, please contact Red Hat Support for assistance.

Report a bug

3.5. INSTALL JBOSS DATA GRID

Prerequisite

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

Procedure 3.3. 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

3.6. RED HAT DOCUMENTATION SITE

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

Report a bug

CHAPTER 4. INSTALL AND USE THE MAVEN REPOSITORY

4.1. 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.

Report a bug

4.2. DOWNLOAD MAVEN

Visit the Maven Download Page for instructions for downloading and installing Maven.

Report a bug

4.3. ABOUT THE JBOSS DATA GRID MAVEN REPOSITORY

Maven is a build tool that creates a standard way of building projects in JBoss Data Grid and helps ensure that codes and examples shipped with the product behave as expected.

Report a bug

4.4. INSTALL THE MAVEN REPOSITORY

4.4.1. Maven Repository Installation Options

There are three ways to install the JBoss Data Grid Maven repository:

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

4.4.2. Local File System Repository Installation

This option is best suited for initial testing in a small team. Use the following command to extract the JBoss Data Grid Maven repository to a directory in your local file system:

Procedure 4.1. Local File System Repository Installation

1. Download the **`jboss-datagrid-maven-repository-6.0.0.zip`** file from the Red Hat Customer Portal.
2. Unzip the file to a directory on your local file system such as **`$JDG_HOME/projects/maven-repositories/`**

Report a bug

4.4.3. Apache Web Server Installation

This option is best suited for use in a multi-user environment. The JBoss Data Grid Maven repository can be installed in a standard webserver (such as Apache httpd) as follows:

Procedure 4.2. Apache Web Server Installation

1. Download the following file from the Red Hat Customer Portal:

   ```
   jboss-datagrid-maven-repository-6.0.0.zip
   ```

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

3. Configure Apache to allow read access and directory browsing in the configured directory.

Report a bug

4.4.4. Maven Repository Manager Installation

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

The JBoss Data Grid repository can be installed using a Maven repository manager using its documentation. Examples of such repository managers are:

- Apache Archiva
- JFrog Artifactory
- Sonatype Nexus

Report a bug

4.5. CONFIGURE THE MAVEN REPOSITORY

4.5.1. Configure the Maven Repository

To configure the installed 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. Refer to the Maven documentation for further information about configuring Maven.

Report a bug

4.5.2. Set the Maven Repository Links

The links of the Maven repository depend on the installation option used. Access the provided links in your browser for each installation type.

Table 4.1. Maven Repository Installation Links

Installation Type	Link
File system installation	file://$JDG_HOME/projects/maven-repositories/jboss-datagrid-maven-repository-6.0.0
Apache Web Server installation	http://intranet.acme.com/jboss-datagrid-maven-repository-6.0.0
Nexus Repository Manager	https://intranet.acme.com/nexus/content/repositories/jboss-datagrid-maven-repository-6.0.0

Report a bug

4.5.3. 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 **`settings.xml`** file:

```
<settings>
      ...
      <profiles>
        ...
        <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-maven-repository-
6.0.0</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>jboss-datagrid-repository-group</id>
              <name>JBoss Data Grid Maven Repository</name>
              <url>file:///path/to/repo/jboss-datagrid-maven-repository-
6.0.0</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>
      <activeProfile>jboss-datagrid-repository</activeProfile>
    </activeProfiles>
    ...
</settings>
```

Report a bug

CHAPTER 5. CREATE A NEW JBOSS DATA GRID PROJECT

5.1. ADD DEPENDENCIES TO YOUR PROJECT

Set up 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

5.2. ADD A PROFILE TO YOUR PROJECT

To enable the JBoss Maven repository for your project, add a profile to your **pom.xml** file 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-maven-repository-
6.0.0</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-maven-repository-6.0.0</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>
```

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 JBoss Data Grid package to the build classpath.

Report a bug

5.3. RUNNING JBOSS DATA GRID

The simplest way to run JBoss Data Grid is to run the following script:

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

This starts JBoss Data Grid using the configuration defines in the **`standalone.xml`** file (located at **`$JDG_HOME/standalone/configuration`**).

Alternate configurations can be run by appending **-c**, followed by the configuration file name. As an example:

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

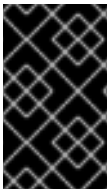
Report a bug

CHAPTER 6. RUN JBOSS DATA GRID AS AN EMBEDDED CACHE

6.1. QUICKSTART FILE LOCATIONS

The exercises detailed in the *Run JBoss Data Grid as an Embedded Cache* and *Run JBoss Data Grid as an Embedded Data Grid* chapters requires specific quickstart files. The required clustered quickstart zip files are available here:

- The Infinispan Github Repository.



IMPORTANT

These quickstarts are written for the Infinispan community project. To run these quickstarts in JBoss Data Grid, replace Infinispan dependencies with JBoss Data Grid dependencies.

Report a bug

6.2. CREATE A MAIN METHOD IN THE QUICKSTART CLASS

Create a new Quickstart class by following the outlined steps:

Procedure 6.1. 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:

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

package com.mycompany.app;
public class Quickstart {
  public static void main(String args[]) throws Exception {
       Cache<Object, Object> c = 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

6.3. CREATE A NEW REMOTECACHEMANAGER

Use the following configuration to declaratively configure a new **`RemoteCacheManager`**:

```
Properties props = new Properties();
props.put("infinispan.client.hotrod.server_list", "127.0.0.1:11222");
RemoteCacheManager manager = new RemoteCacheManager(props);
RemoteCache defaultCache = manager.getCache();
```

NOTE

To learn more about using **`Hot Rod`** with JBoss Data Grid, refer to the *Developer Guide*'s Hot Rod Chapter.

Report a bug

6.4. USE THE DEFAULT CACHE

6.4.1. Add and Remove Data from the Cache

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 6.2. 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);
```

Report a bug

6.4.2. Adding and Replacing a Key Value

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 6.3. Adding and Replacing a Key Value

1. Add an entry with **`key`** as the key value.

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

2. Check if the key value is missing. If the value is not found, replace the absent value with **`key`**. If the value is found, no change occurs.

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

See Also:

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

Report a bug

6.4.3. Adjust Data Life

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 6.4. Adjust the Data Life

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

```
cache.put("key", "value", 5, 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

6.4.4. 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

6.5. USE A NAMED CACHE

6.5.1. Using a Named Cache

A named cache in JBoss Data Grid does not need to be declared in the configuration. If undefined, the named cache is automatically configured in a manner identical to the default cache.

The user can override this behavior by defining the named cache with the required custom configuration either declaratively (using XML) or programmatically (using the API).

If the custom cache is registered declaratively, the configuration file must be loaded before it can be used. If the custom cache is registered programmatically, no configuration file loading is required.

Report a bug

6.5.2. Register the Named Cache Programmatically

JBoss Data Grid offers a fluent configuration feature that allows users to programmatically register named caches.

The named cache can be configured programmatically. The **`CustomCacheQuickstart.java`** file included in the Quickstart package can be referred to as an example of this. The infinispan-quickstart can be found in **`infinispan-quickstart/embedded-cache/src/main/java/org/infinispan/quickstart/embeddedcache/`**.

See Also:

- Section 6.1, “Quickstart File Locations”

Report a bug

6.5.3. Load the Configuration File

Prerequisites:

Register the named cache declaratively.

The **`XmlConfiguredCacheQuickstart.java`** file in the infinispan-quickstarts demonstrates how the custom configuration of the named cache is loaded. This file is located in **`infinispan-quickstart/embedded-cache/src/main/java/org/infinispan/quickstart/embeddedcache`** in the Quickstarts package.

See Also:

- Section 6.1, “Quickstart File Locations”

Report a bug

6.5.4. 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 **`infinispan-quickstart/embedded-cache/src/main/resources`** in the infinispan-quickstarts package.

See Also:

- Section 6.1, “Quickstart File Locations”

Report a bug

CHAPTER 7. RUN JBOSS DATA GRID AS AN EMBEDDED DATA GRID

7.1. QUICKSTART FILE LOCATIONS

The exercises detailed in the *Run JBoss Data Grid as an Embedded Cache* and *Run JBoss Data Grid as an Embedded Data Grid* chapters requires specific quickstart files. The required clustered quickstart zip files are available here:

- The Infinispan Github Repository.

IMPORTANT

These quickstarts are written for the Infinispan community project. To run these quickstarts in JBoss Data Grid, replace Infinispan dependencies with JBoss Data Grid dependencies.

Report a bug

7.2. RUN JBOSS DATA GRID AS AN EMBEDDED DATA GRID

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.

Report a bug

7.3. SHARING JGROUP CHANNELS

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

7.4. RUN JBOSS DATA GRID IN A CLUSTER

7.4.1. Compile the Project

Use Maven to compile your project with the following command:

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

Report a bug

7.4.2. Run the Clustered Cache with Replication Mode

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

Procedure 7.1. 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

7.4.3. Run the Clustered Cache with Distribution Mode

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

Procedure 7.2. 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

7.4.4. Configure the Cluster

7.4.4.1. Configuring the Cluster

Use the following steps to add and configure your cluster:

Procedure 7.3. Configure the Cluster

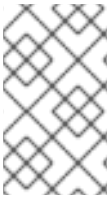
1. Add the default configuration for a new cluster
2. Customize the default cluster configuration according to the requirements of your network. This can be done declaratively (using XML) or programmatically.
3. Configure the replicated or distributed data grid.

Report a bug

7.4.4.2. Add the Default Cluster Configuration

Add a cluster configuration to ensure that 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

7.4.4.3. Customize the Default Cluster Configuration

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

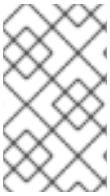
Programmatic Configuration:

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 JBoss Data Grid's XML configuration:

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

Report a bug

7.4.4.4. Configure the Replicated Data Grid

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 **`cfg.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");
}
```

Report a bug

7.4.4.5. Configure the Distributed Data Grid

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 8. RUN A JBOSS DATA GRID AS A NODE WITHOUT ENDPOINTS

8.1. ABOUT 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

8.2. BENEFITS OF A NODE WITHOUT ENDPOINTS

The primary benefit for creating a node without endpoints in 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

8.3. SAMPLE CONFIGURATION FOR A NODE WITHOUT ENDPOINTS

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

Procedure 8.1. 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

8.4. 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

CHAPTER 9. JBOSS DATA GRID CARMART QUICKSTARTS

9.1. ABOUT THE CARMART QUICKSTART

The 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 programmatically and run in the same Java Virtual Machine (JVM) as the web application.

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-quickstarts-1.0.0/carmart/`

Report a bug

9.2. ABOUT THE CARMART TRANSACTIONAL QUICKSTART

The transactional version of 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 programmatically 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 Tomcat.

Location

JBoss Data Grid's Transactional CarMart Quickstart can be found at the following location: `jboss-datagrid-quickstarts-1.0.0/carmart-transactional`

Report a bug

9.3. DIFFERENCES BETWEEN THE CARMART AND TRANSACTIONAL QUICKSTARTS

Despite the similarity in steps to build, deploy and undeploy 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

9.4. THE CARMART QUICKSTART USING JBOSS APPLICATION SERVER

9.4.1. Build the CarMart Quickstart to the JBoss Application Server

The following procedure provides directions to build the CarMart application to JBoss Application Server 7.

Prerequisite

Obtain the supported JBoss Data Grid Library Mode distribution files.

Procedure 9.1. Build CarMart to JBoss Application Server

1. **Install the Maven Repository**
   For initial testing in a small team, extract the JBoss Data Grid Maven repository to a directory on the local system:

   ```
   $ unzip jboss-datagrid-maven-repository-6.0.0.zip
   ```

   For further information about the installation methods available to install the JBoss Data Grid Maven repository, refer to Chapter 4, *Install and Use the Maven Repository*.

2. **Start JBoss Application Server**
   Use the following to start the JBoss Application Server:

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

3. **Build your Application**
   Use the following command to build your application using Maven:

```
$ mvn clean package -Plibrary-jbossas -
Ddatagrid.maven.repo=file:///path/to/unpacked/jdg/maven/repository
```

Report a bug

9.4.2. Deploy the CarMart Quickstart to the JBoss Application Server

The following procedure outlines how to deploy the CarMart quickstart in JBoss Application Server 7:

Prerequisite

The CarMart quickstart must be built to be deployed.

Procedure 9.2. Deploy the CarMart Quickstart to the JBoss Application Server

1. **Deploy your Application**
   Use JBoss Application Server's Maven plugin to deploy the application as follows:

```
$ mvn jboss-as:deploy -Plibrary-jbossas
```

2. **View your Application**
   To view the application, use your browser to navigate to the following link:

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

Report a bug

9.4.3. Undeploy the CarMart Quickstart from JBoss Application Server

The following procedure provides directions to undeploy an already deployed application from JBoss Application Server.

Procedure 9.3. Undeploy an Application from JBoss Application Server

- To undeploy an application, use the following command:

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

Report a bug

9.5. THE CARMART QUICKSTART USING TOMCAT

9.5.1. Build the CarMart Quickstart to Tomcat

The following procedure provides directions to build the CarMart quickstart to Tomcat 7.

Prerequisite

Obtain the supported JBoss Data Grid Library Mode distribution files.

Procedure 9.4. Build the CarMart Quickstart to Tomcat

1. **Install the Maven Repository**
   For initial testing in a small team, extract the JBoss Data Grid Maven repository to a directory on the local system:

   ```
   $ unzip jboss-datagrid-maven-repository-6.0.0.zip
   ```

   For further information about the installation methods available to install the JBoss Data Grid Maven repository, refer to Chapter 4, *Install and Use the Maven Repository*.

2. **Add Manager Script Information**
   This build assumes that the default configuration for **Tomcat 7** will be used, including a default host name (**`localhost`**) and port (**`8080`**). To allow **Maven**'s **Tomcat** plug-in to access the manager, add the following information to the **`tomcat-users.xml`** file (located at **`conf/tomcat-users.xml`**):

   ```
   <role rolename="manager-script"/>
   <user username="admin" password="" roles="manager-script"/>
   ```

3. **Start Tomcat**
   Use the following script to start Tomcat:

   ```
   $CATALINA_HOME/bin/catalina.sh start
   ```

4. **Build your Application**
   Use the following command to build your application using Maven:

   ```
   $ mvn clean package -Plibrary-jbossas -
   Ddatagrid.maven.repo=file:///path/to/unpacked/jdg/maven/repository
   ```

Report a bug

9.5.2. Deploy the CarMart Quickstart to Tomcat

The following procedure outlines how to deploy the CarMart quickstart to Tomcat 7:

Prerequisite

The CarMart quickstart must be built to be deployed.

Procedure 9.5. Deploy the CarMart Quickstart to Tomcat

1. **Edit Server Credentials**
   Add the **`server`** element in your Maven **`settings.xml`** file (located in your Maven installation's **`conf`** directory) with the required credentials:

   ```
   <server>
        <id>tomcat</id>
        <username>admin</username>
        <password></password>
    </server>
   ```

2. **Deploy your Application**
   Use Tomcat's Maven plugin to deploy the application as follows:

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

3. **View your Application**
   To view the application, use your browser to navigate to the following link:

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

Report a bug

9.5.3. Undeploy the CarMart Quickstart from Tomcat

The following procedure provides directions to undeploy an already deployed application from Tomcat.

Procedure 9.6. Undeploy an Application from Tomcat

- To undeploy an application, use the following command:

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

Report a bug

9.6. THE CARMART QUICKSTART IN REMOTE CLIENT-SERVER MODE

9.6.1. Build the CarMart Quickstart in Remote Client-Server Mode

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

Prerequisite

Obtain the supported JBoss Data Grid Library Mode distribution files.

Procedure 9.7. Build 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 after the closing tags for the <***`system-properties`***> element:

   ```
   <paths>
   <path name="temp" path="/tmp"/>
   </paths>
   ```

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

   ```
   <local-cache name="carcache"
   ```

```
            start="EAGER" batching="false"
            indexing="NONE">
    <locking isolation="REPEATABLE_READ"
      striping="false"
      acquire-timeout="20000"
            concurrency-level="500"/>
    <eviction strategy="LIRS"
             max-entries="4"/>
    <file-store relative-to="temp"
                path="carstore"
                passivation="false"/>
</local-cache>
```

2. **Start the JBoss Data Grid Server**
   Run the following script to start the JBoss Data Grid Server:

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

NOTE

Using the provided configurations, the server runs on the **`test1`** address.

   - **Configure test1 for UNIX Users**
     Configure the **`test1`** address on a UNIX system by adding the following line to **`/etc/hosts`**:

```
192.168.11.101    test1
```

   - **Run the IfConfig Command**
     Run the following command on the command line:

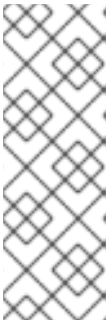
```
$ sudo ifconfig eth0:1 192.168.11.101 netmask 255.255.255.0
```

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

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

4. **Specify the JBoss Data Grid Server Address**
   Edit the **`jdg.properties`** file (located in the **`$JDG_HOME/src/main/resources/META-INF/`** directory) to specify the address of the JBoss Data Grid server as follows:

```
datagrid.address=test1
```



NOTE

If the JBoss Data Grid server is running on a **`localhost`** address with defined port-offset, modify the **`jdg.properties`** file to include the port information. For example:

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

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

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

Report a bug

9.6.2. Deploy the CarMart Quickstart in Remote Client-Server Mode

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

Prerequisite

The CarMart quickstart must be built to be deployed.

Procedure 9.8. Deploy the CarMart Quickstart in Remote Client-Server Mode

1. **Deploy your Application**
   Deploy your application using Maven with the following command:

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

2. **View your Application**
   Visit the following link to view the application:

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

Report a bug

9.6.3. Undeploy the CarMart Quickstart in Remote Client-Server Mode

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

Procedure 9.9. Undeploy an Application in Remote Client-Server Mode

- To undeploy an application, use the following command:

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

Report a bug

CHAPTER 10. FOOTBALL QUICKSTART ENDPOINT EXAMPLES

10.1. ABOUT FOOTBALL QUICKSTART ENDPOINT EXAMPLES

The Football application is a simple example to illustrate the use of 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-quickstarts-1.0.0/carmart/rest-endpoint`**
- **`jboss-datagrid-quickstarts-1.0.0/carmart/hotrod-endpoint`**
- **`jboss-datagrid-quickstarts-1.0.0/carmart/memcached-endpoint`**

Report a bug

10.2. 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 JBoss Data Grid.

NOTE

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

Procedure 10.1. 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:

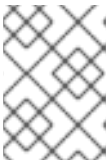
   ```
   <subsystem xmlns="urn:jboss:domain:datasources:1.0">
   ```

```
 <datasources>
  <datasource jndi-name="java:jboss/datasources/ExampleDS"
        pool-name="ExampleDS"
        enabled="true"
        use-java-context="true">
   <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-
1</connection-url>
   <driver>h2</driver>
   <security>
    <user-name>sa</user-name>
    <password>sa</password>
   </security>
  </datasource>
  <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:jboss:domain:infinispan:1.2" default-cache-
container="local">
 <cache-container name="local"
    default-cache="memcachedCache"
    listener-executor="infinispan-listener"
    start="EAGER">
  <local-cache name="memcachedCache"
        start="EAGER"
        batching="false"
        indexing="NONE">
   <locking isolation="REPEATABLE_READ"
     acquire-timeout="20000"
     concurrency-level="500"
     striping="false" />
   <transaction mode="NONE" />
   <string-keyed-jdbc-store
datasource="java:jboss/datasources/ExampleDS"
       passivation="false"
       preload="false"
        purge="false">
    <property name="databaseType">H2</property>
    <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>
  <local-cache name="teams"
        start="EAGER"
          batching="false"
        indexing="NONE">
   <locking isolation="REPEATABLE_READ"
     acquire-timeout="20000"
      concurrency-level="500"
     striping="false" />
   <transaction mode="NONE" />
   <string-keyed-jdbc-store
datasource="java:jboss/datasources/ExampleDS"
       passivation="false"
       preload="false"
       purge="false">
    <property name="databaseType">H2</property>
    <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>
 </cache-container>
</subsystem>
```



NOTE

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

2. **Edit the Submodule Configuration File**
   Each submodule (specifically **`hotrod-endpoint`**, **`rest-endpoint`** and **`memcached-endpoint`**) contains a configuration file (located at **`$JDG_HOME/src/main/resources/jdg.properties`**). Edit the configuration to specify the values required for your JBoss Data Grid installation. The Hot Rod endpoint requires further configuration, as follows:

   a. **Install the Maven Repository**
      For initial testing in a small team, the repository can be extracted to a directory on the local file system as follows:

      ```
      unzip jboss-datagrid-maven-repository-6.0.0.zip
      ```

      This creates a Maven repository in a directory called **`jboss-datagrid-maven-repository-6.0.0`**.

      For further information about the installation methods available to install the JBoss Data Grid Maven repository, refer to Chapter 4, *Install and Use the Maven Repository*.

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

   ```
   mvn 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.

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

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

5. **Run the Application**
   Run the example application in its directory.

   a. For the Hot Rod endpoint, run the following command:

      ```
      java -jar target/hotrod-endpoint-quickstart.jar
      ```

   b. For the memcached endpoint, run the following command:

      ```
      java -jar target/memcached-endpoint-quickstart.jar
      ```

   c. For the REST endpoint, run the following command:

      ```
      java -jar target/rest-endpoint-quickstart.jar
      ```

Report a bug

CHAPTER 11. REMOVE JBOSS DATA GRID

11.1. REMOVE JBOSS DATA GRID FROM YOUR LINUX SYSTEM

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

WARNING

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

Procedure 11.1. 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 name of your JBoss Data Grid home directory:

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

Report a bug

11.2. REMOVE JBOSS DATA GRID FROM YOUR WINDOWS SYSTEM

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

WARNING

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

Procedure 11.2. 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. REVISION HISTORY

Revision 0.33-4.400 **2013-10-31** **Rüdiger Landmann**

Rebuild with publican 4.0.0

Revision 0.33-4 **Tue Aug 06 2013** **Misha Husnain Ali**

Updated with new product name.

Revision 0.33-3 **Tue Aug 06 2013** **Gemma Sheldon**

Updated Quickstart File Locations with Important info.

Built from Content Specification: 7679, Revision: 194247 by gsheldon