



Red Hat JBoss Data Grid 6.1

Getting Started Guide

For use with Red Hat JBoss Data Grid 6.1

Edition 2

Red Hat JBoss Data Grid 6.1 Getting Started Guide

For use with Red Hat JBoss Data Grid 6.1
Edition 2

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

Copyright © 2013 Red Hat, Inc.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This guide outlines introductory concepts and operations within Red Hat JBoss Data Grid 6.1

Table of Contents

PREFACE	4
PART I. INTRODUCING JBOSS DATA GRID	5
CHAPTER 1. JBOSS DATA GRID	6
1.1. ABOUT JBOSS DATA GRID	6
1.2. JBOSS DATA GRID SUPPORTED CONFIGURATIONS	6
1.3. JBOSS DATA GRID USAGE MODES	6
1.3.1. JBoss Data Grid Usage Modes	6
1.3.2. Remote Client-Server Mode	6
1.3.3. Library Mode	7
1.4. JBOSS DATA GRID BENEFITS	7
1.5. JBOSS DATA GRID VERSION INFORMATION	9
1.6. JBOSS DATA GRID CACHE ARCHITECTURE	9
1.7. JBOSS DATA GRID APIS	10
PART II. DOWNLOAD AND INSTALL JBOSS DATA GRID	12
CHAPTER 2. PREREQUISITES FOR JBOSS DATA GRID	13
2.1. JBOSS DATA GRID PREREQUISITES	13
2.2. JAVA VIRTUAL MACHINE	13
2.3. INSTALL OPENJDK ON RED HAT ENTERPRISE LINUX	13
CHAPTER 3. DOWNLOAD AND INSTALL JBOSS DATA GRID	14
3.1. DOWNLOAD JBOSS DATA GRID	14
3.2. ABOUT THE RED HAT CUSTOMER PORTAL	14
3.3. CHECKSUM VALIDATION	14
3.4. VERIFY THE DOWNLOADED FILE	15
3.5. INSTALL JBOSS DATA GRID	15
3.6. RED HAT DOCUMENTATION SITE	16
CHAPTER 4. INSTALL AND USE THE MAVEN REPOSITORIES	17
4.1. ABOUT MAVEN	17
4.2. REQUIRED MAVEN REPOSITORIES	17
4.3. INSTALL THE MAVEN REPOSITORIES	17
4.3.1. Maven Repository Installation Options	17
4.3.2. Local File System Repository Installation	17
4.3.3. Apache Web Server Installation	18
4.3.4. Maven Repository Manager Installation	19
4.4. CONFIGURE THE MAVEN REPOSITORIES	19
4.4.1. Configure the Maven Repository	19
4.4.2. Set the Maven Repository Links	19
4.4.3. Next Steps	20
PART III. RUN JBOSS DATA GRID IN REMOTE CLIENT-SERVER MODE	21
CHAPTER 5. RUN JBOSS DATA GRID IN REMOTE CLIENT-SERVER MODE	22
5.1. PREREQUISITES	22
5.2. RUN JBOSS DATA GRID IN STANDALONE MODE	22
5.3. RUN JBOSS DATA GRID IN CLUSTERED MODE	22
5.4. RUN JBOSS DATA GRID WITH A CUSTOM CONFIGURATION	22
5.5. SET AN IP ADDRESS TO RUN JBOSS DATA GRID	23
PART IV. RUN JBOSS DATA GRID IN LIBRARY MODE	24

CHAPTER 6. CREATE A NEW JBOSS DATA GRID PROJECT	25
6.1. ADD DEPENDENCIES TO YOUR PROJECT	25
6.2. ADD A PROFILE TO YOUR PROJECT	25
6.3. RUNNING JBOSS DATA GRID	26
CHAPTER 7. RUN JBOSS DATA GRID AS AN EMBEDDED CACHE (LIBRARY MODE)	28
7.1. QUICKSTART FILE LOCATIONS	28
7.2. CREATE A MAIN METHOD IN THE QUICKSTART CLASS	28
7.3. CREATE A NEW REMOTECACHEMANAGER	29
7.4. USE THE DEFAULT CACHE	29
7.4.1. Add and Remove Data from the Cache	29
7.4.2. Adding and Replacing a Key Value	30
7.4.3. Adjust Data Life	30
7.4.4. Default Data Mortality	30
7.5. USE A NAMED CACHE	31
7.5.1. Using a Named Cache	31
7.5.2. Register the Named Cache Programmatically	31
7.5.3. Load the Configuration File	31
7.5.4. Register the Named Cache Using XML	31
CHAPTER 8. RUN JBOSS DATA GRID IN LIBRARY MODE	33
8.1. QUICKSTART FILE LOCATIONS	33
8.2. RUN JBOSS DATA GRID AS AN EMBEDDED DATA GRID (LIBRARY MODE)	33
8.3. SHARING JGROUP CHANNELS	33
8.4. RUN JBOSS DATA GRID IN A CLUSTER	33
8.4.1. Compile the Project	33
8.4.2. Run the Clustered Cache with Replication Mode	34
8.4.3. Run the Clustered Cache with Distribution Mode	34
8.4.4. Configure the Cluster	35
8.4.4.1. Configuring the Cluster	35
8.4.4.2. Add the Default Cluster Configuration	35
8.4.4.3. Customize the Default Cluster Configuration	35
8.4.4.4. Configure the Replicated Data Grid	36
8.4.4.5. Configure the Distributed Data Grid	37
CHAPTER 9. RUN A JBOSS DATA GRID AS A NODE WITHOUT ENDPOINTS	38
9.1. ABOUT ENDPOINTS	38
9.2. BENEFITS OF A NODE WITHOUT ENDPOINTS	38
9.3. SAMPLE CONFIGURATION FOR A NODE WITHOUT ENDPOINTS	38
9.4. CONFIGURE A NODE WITH NO ENDPOINTS	38
CHAPTER 10. MONITOR JBOSS DATA GRID APPLICATIONS IN JBOSS ENTERPRISE APPLICATION PLATFORM	40
10.1. ABOUT JBOSS DATA GRID APPLICATIONS FOR JBOSS ENTERPRISE APPLICATION PLATFORM	40
10.2. PREREQUISITES	40
10.3. MONITOR JBOSS DATA GRID APPLICATIONS IN JBOSS ENTERPRISE APPLICATION PLATFORM	40
PART V. JBOSS DATA GRID QUICKSTARTS	42
CHAPTER 11. THE HELLO WORLD QUICKSTART	43
11.1. ABOUT THE HELLO WORLD QUICKSTART	43
11.2. QUICKSTART PREREQUISITES	43
11.3. START TWO APPLICATION SERVER INSTANCES	43
11.4. BUILD AND DEPLOY THE HELLO WORLD QUICKSTART	44
11.5. ACCESS THE RUNNING APPLICATION	45

11.6. TEST REPLICATION ON THE APPLICATION	45
11.7. REMOVE THE APPLICATION	46
11.8. DEBUG THE APPLICATION	47
CHAPTER 12. THE CARMART QUICKSTARTS	48
12.1. ABOUT THE CARMART QUICKSTART	48
12.2. ABOUT THE CARMART TRANSACTIONAL QUICKSTART	48
12.3. DIFFERENCES BETWEEN THE CARMART AND TRANSACTIONAL QUICKSTARTS	49
12.4. QUICKSTART PREREQUISITES	49
12.5. THE CARMART QUICKSTART USING A JBOSS SERVER	50
12.5.1. Build and Deploy the CarMart Quickstart to the JBoss Server	50
12.5.2. View the CarMart Quickstart on the JBoss Application Server	50
12.5.3. Remove the CarMart Quickstart from the JBoss Server	51
12.6. THE CARMART QUICKSTART TO A SERVER	51
12.6.1. Build and Deploy the CarMart Quickstart to the Server	51
12.6.2. View the CarMart Quickstart on the Server	52
12.6.3. Remove the CarMart Quickstart from the Server	52
12.7. THE CARMART QUICKSTART IN REMOTE CLIENT-SERVER MODE	53
12.7.1. Build and Deploy the CarMart Quickstart in Remote Client-Server Mode	53
12.7.2. View the CarMart Quickstart in Remote Client-Server Mode	54
12.7.3. Remove the CarMart Quickstart in Remote Client-Server Mode	55
CHAPTER 13. THE FOOTBALL QUICKSTART ENDPOINT EXAMPLES	56
13.1. ABOUT FOOTBALL QUICKSTART ENDPOINT EXAMPLES	56
13.2. QUICKSTART PREREQUISITES	56
13.3. BUILD THE FOOTBALL APPLICATION	56
PART VI. UNINSTALL JBOSS DATA GRID	61
CHAPTER 14. REMOVE JBOSS DATA GRID	62
14.1. REMOVE JBOSS DATA GRID FROM YOUR LINUX SYSTEM	62
14.2. REMOVE JBOSS DATA GRID FROM YOUR WINDOWS SYSTEM	62
APPENDIX A. REFERENCES	64
A.1. ABOUT KEY-VALUE PAIRS	64
APPENDIX B. MAVEN CONFIGURATION INFORMATION	65
B.1. INSTALL THE JBOSS ENTERPRISE APPLICATION PLATFORM REPOSITORY USING NEXUS	65
B.2. MAVEN REPOSITORY CONFIGURATION EXAMPLE	66
APPENDIX C. REVISION HISTORY	68

PREFACE

PART I. INTRODUCING JBOSS DATA GRID

CHAPTER 1. JBOSS DATA GRID

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 the data grid using REST, Memcached, Hot Rod, or simple map-like API.

JBoss Data Grid 6.1 and JBoss Data Grid 6.1 Beta is based on Infinispan version 5.2.

[Report a bug](#)

1.2. JBOSS DATA GRID SUPPORTED CONFIGURATIONS

The set of supported features, configurations, and integrations for JBoss Data Grid (6.0 and 6.1) are available at the Supported Configurations page at <https://access.redhat.com/knowledge/articles/115883>.

[Report a bug](#)

1.3. JBOSS DATA GRID USAGE MODES

1.3.1. JBoss Data Grid Usage Modes

JBoss Data Grid offers two usage modes:

- Remote Client-Server mode
- Library mode

[Report a bug](#)

1.3.2. 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 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.

However, there are a number of benefits to running JBoss Data Grid in Remote Client-Server mode if you do not require any features that require Library mode. 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.

[Report a bug](#)

1.3.3. Library Mode

Library mode allows the user to build and deploy a custom runtime environment. The Library usage mode hosts a single data grid node in the applications process, with remote access to nodes hosted in other JVMs. Tested containers for JBoss Data Grid 6 Library mode includes Tomcat 7 and JBoss Enterprise Application Platform 6. Library mode is supported in JBoss Data Grid 6.0.1.

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

Use Library mode if you require:

- transactions.
- listeners and notifications.

[Report a bug](#)

1.4. JBOSS DATA GRID BENEFITS

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 carried 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 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. 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. 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](#)

1.6. JBOSS DATA GRID CACHE ARCHITECTURE

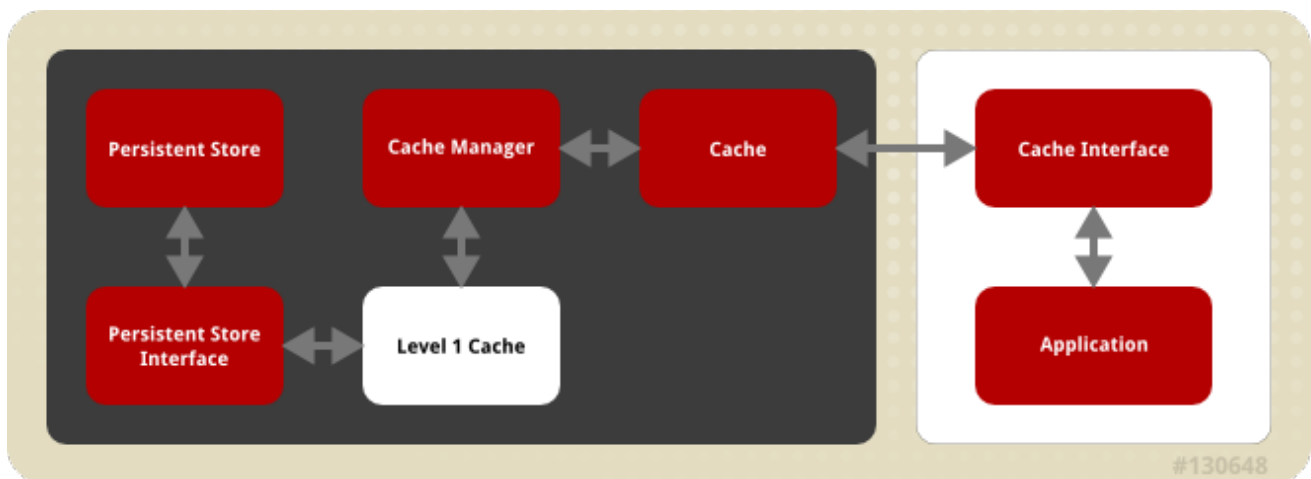


Figure 1.1. JBoss Data Grid Cache Architecture

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 stores 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 stores 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](#)

1.7. JBOSS DATA GRID APIS

JBoss Data Grid provides the following programmable APIs:

- Cache
- Batching
- Grouping
- CacheStore and 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 JBOSS DATA GRID

CHAPTER 2. PREREQUISITES FOR JBOSS DATA GRID

2.1. JBOSS DATA GRID PREREQUISITES

The only prerequisites to set up JBoss Data Grid is a Java Virtual Machine (compatible with Java 6.0 or better) and that the most recent supported version of the product is installed on your system.

[Report a bug](#)

2.2. JAVA VIRTUAL MACHINE

A Java Virtual Machine (JVM) creates a virtual environment where the Java bytecode can 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](#)

2.3. 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 root 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](#)

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 {VERSION}** for JBoss Data Grid with the Remote Client-Server usage mode or **JBoss Data Grid Library {VERSION}** 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 the centralized platform for Red Hat knowledge and subscription resources. It allows you 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](#)

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 REPOSITORIES

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.



IMPORTANT

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

Visit the [Maven Download Page](#) for instructions for downloading and installing Maven.

[Report a bug](#)

4.2. REQUIRED MAVEN REPOSITORIES

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](#)

4.3. INSTALL THE MAVEN REPOSITORIES

4.3.1. Maven Repository Installation Options

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](#)

4.3.2. Local File System Repository Installation

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

Procedure 4.1. 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-maven-repository-{VERSION}.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/`

Procedure 4.2. Local File System Repository Installation (JBoss Enterprise Application Platform)

1. Log Into the Customer Service Portal

In a browser window, navigate to the [software page](#) for JBoss Enterprise Application Platform.

2. Locate the Correct Entry

Find "Application Platform 6 Maven Repository" in the list.

3. Download the Repository

Click the **Download** button to download a zip file containing the repository.

4. Unzip the Repository File

Unzip the file in the same directory on the local file system into `$JDG_HOME/projects/maven-repositories/`.

[Report a bug](#)

4.3.3. 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 4.3. Apache Web Server Installation (JBoss Data Grid)

1. Log Into the Customer Service Portal

In a browser window, navigate to <https://access.redhat.com/home> (Red Hat Customer Service Portal) and log in.

2. Download the JBoss Data Grid Repository File

Download the following file from the Red Hat Customer Portal:

```
jboss-datagrid-maven-repository-{VERSION}.zip
```

3. Unzip the Repository Files

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

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

Procedure 4.4. Apache Web Server Installation (JBoss Enterprise Application Platform)

1. Log Into the Customer Service Portal

In a browser window, navigate to the JBoss Enterprise Application Platform software page at <https://access.redhat.com/jbossnetwork/restricted/listSoftware.html?product=appplatform>

2. Locate the Correct Entry

Find "Application Platform 6 Maven Repository" in the list.

3. Download the Repository

Click the **Download** button to download a zip file containing the repository.

4. Unzip the Repository File

Unzip the file in the same directory on the local file system into `$JDG_HOME/projects/maven-repositories/`.

5. Configure Apache

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

[Report a bug](#)

4.3.4. Maven Repository Manager Installation

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

The JBoss Data Grid and JBoss Enterprise Application Server repositories can be 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, refer to [Section B.1, "Install the JBoss Enterprise Application Platform Repository Using Nexus"](#)

[Report a bug](#)

4.4. CONFIGURE THE MAVEN REPOSITORIES

4.4.1. 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 <http://maven.apache.org/settings.html> (the Maven documentation) for more information about configuring Maven.

Refer to [Section B.2, "Maven Repository Configuration Example"](#) to view a sample Maven configuration.

[Report a bug](#)

4.4.2. 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-{VERSION}
Apache Web Server installation	http://intranet.acme.com/jboss-datagrid-maven-repository-{VERSION}
Nexus Repository Manager	https://intranet.acme.com/nexus/content/repositories/jboss-datagrid-maven-repository-{VERSION}

[Report a bug](#)

4.4.3. Next Steps

After the newest available version of JBoss Data Grid is installed and Maven is set up and configured, refer to [Chapter 6, *Create a New JBoss Data Grid Project*](#) to learn how to use JBoss Data Grid for the first time.

[Report a bug](#)

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

CHAPTER 5. RUN JBOSS DATA GRID IN REMOTE CLIENT-SERVER MODE

5.1. PREREQUISITES

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

- Ensure an appropriate version of OpenJDK is installed. For more information, refer to [Section 2.3, “Install OpenJDK on Red Hat Enterprise Linux”](#)
- Download and install the latest version of JBoss Data Grid. For more information, refer to [Section 3.1, “Download JBoss Data Grid”](#)

[Report a bug](#)

5.2. RUN JBOSS DATA GRID IN STANDALONE MODE

Standalone mode refers to a single instance of 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](#)

5.3. RUN JBOSS DATA GRID IN CLUSTERED MODE

Clustered mode refers to a cluster made up of two or more 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](#)

5.4. RUN JBOSS DATA GRID WITH A CUSTOM CONFIGURATION

To run 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](#)

5.5. SET AN IP ADDRESS TO RUN JBOSS DATA GRID

For production use, the 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](#)

PART IV. RUN JBOSS DATA GRID IN LIBRARY MODE

CHAPTER 6. CREATE A NEW JBOSS DATA GRID PROJECT

6.1. 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](#)

6.2. 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-
{VERSION}</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-
{VERSION}</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](#)

6.3. RUNNING 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
```

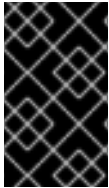
[Report a bug](#)

CHAPTER 7. RUN JBOSS DATA GRID AS AN EMBEDDED CACHE (LIBRARY MODE)

7.1. QUICKSTART FILE LOCATIONS

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

- The Infinispan Github Repository: <https://github.com/infinispan/infinispan-quickstart>



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. CREATE A MAIN METHOD IN THE QUICKSTART CLASS

Create a new Quickstart class by following the outlined steps:

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:

```
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:

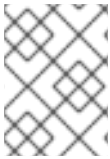

```
$ java -cp target/classes/:target/dependency/*
com.mycompany.app.Quickstart
```

[Report a bug](#)

7.3. 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](#)

7.4. USE THE DEFAULT CACHE

7.4.1. 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 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](#)

7.4.2. 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 7.3. 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](#)

7.4.3. 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 7.4. 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](#)

7.4.4. 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](#)

7.5. USE A NAMED CACHE

7.5.1. 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](#)

7.5.2. 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 7.1, “Quickstart File Locations”](#)

[Report a bug](#)

7.5.3. 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 7.1, “Quickstart File Locations”](#)

[Report a bug](#)

7.5.4. 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 7.1, “Quickstart File Locations”](#)

[Report a bug](#)

CHAPTER 8. RUN JBOSS DATA GRID IN LIBRARY MODE

8.1. QUICKSTART FILE LOCATIONS

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

- The Infinispan Github Repository: <https://github.com/infinispan/infinispan-quickstart>



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](#)

8.2. RUN JBOSS DATA GRID AS AN EMBEDDED DATA GRID (LIBRARY MODE)

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](#)

8.3. 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](#)

8.4. RUN JBOSS DATA GRID IN A CLUSTER

8.4.1. Compile the Project

Use Maven to compile your project with the following command:

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

[Report a bug](#)

8.4.2. 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 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](#)

8.4.3. 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 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](#)

8.4.4. Configure the Cluster

8.4.4.1. Configuring the Cluster

Use the following steps to add and configure your cluster:

Procedure 8.3. Configure the Cluster

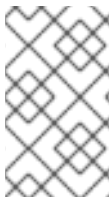
1. Add the default configuration for a new cluster
2. Customize the default cluster configuration according to the requirements of your network. This can be done declaratively (using XML) or programmatically.
3. Configure the replicated or distributed data grid.

[Report a bug](#)

8.4.4.2. 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](#)

8.4.4.3. 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](#)

8.4.4.4. 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](#)

8.4.4.5. 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 9. RUN A JBOSS DATA GRID AS A NODE WITHOUT ENDPOINTS

9.1. 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](#)

9.2. 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](#)

9.3. 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 9.1. 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](#)

9.4. 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 10. MONITOR JBOSS DATA GRID APPLICATIONS IN JBOSS ENTERPRISE APPLICATION PLATFORM

10.1. ABOUT JBOSS DATA GRID APPLICATIONS FOR JBOSS ENTERPRISE APPLICATION PLATFORM

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](#)

10.2. PREREQUISITES

The following are prerequisites to monitor a 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 Beta or better) Library mode plug-in.

[Report a bug](#)

10.3. MONITOR JBOSS DATA GRID APPLICATIONS IN JBOSS ENTERPRISE APPLICATION PLATFORM

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

Procedure 10.1. 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. JBOSS DATA GRID QUICKSTARTS

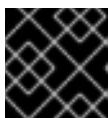
CHAPTER 11. THE HELLO WORLD QUICKSTART

11.1. ABOUT THE HELLO WORLD QUICKSTART

Hello World is a simple quickstart that illustrates how to store data to and retrieve data from a cache using 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 JBoss Application Server 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.

[Report a bug](#)

11.2. 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 JBoss Application Server 7.x.
- Maven 3.0 or better.
- Configure the Maven Repository. For details, refer to [Chapter 4, Install and Use the Maven Repositories](#)

[Report a bug](#)

11.3. 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 JBoss Application Server 7.x). Start two instances of the selected application server as outlines in the following procedures.

Procedure 11.1. 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 11.2. 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:

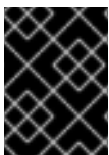
```
$JBOSS_HOME2/bin/standalone.sh -Djboss.socket.binding.port-  
offset=100
```

b. For Windows:

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

[Report a bug](#)

11.4. 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 11.3. 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-as-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-as-helloworld-jdg.war` to the second running server instance.

[Report a bug](#)

11.5. ACCESS THE RUNNING APPLICATION

The Hello World quickstart application runs on the following URLs:

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

[Report a bug](#)

11.6. 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 11.4. Test Replication on the Application

1. Access the First Server

Access the first application server and enter the key and value.

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

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

- Insert the key **foo**.
- Insert the value **bar**.

2. Access the Second Server

Access the second application server and enter the key and value.

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

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

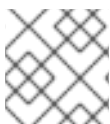
- Click **Get Some**.
- Get the key **foo**.

- 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-as-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-as-helloworld-jdg/TestServletPut
```

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

[Report a bug](#)

11.7. REMOVE THE APPLICATION

Use the following procedure to remove the Hello World application:

Procedure 11.5. 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 -Ddeploy.port=10099
```

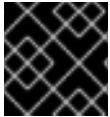
[Report a bug](#)

11.8. 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 12. THE CARMART QUICKSTARTS

12.1. 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 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-quickstarts-6.1/**



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](#)

12.2. 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 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 Tomcat.

Location

JBoss Data Grid's Transactional CarMart Quickstart can be found at the following location: **jboss-datagrid-quickstarts-6.1/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](#)

12.3. 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](#)

12.4. 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 JBoss Application Server 7.x.
- Maven 3.0 or better.
- Configure the Maven Repository. For details, refer to [Chapter 4, Install and Use the Maven Repositories](#)

[Report a bug](#)

12.5. THE CARMART QUICKSTART USING A JBOSS SERVER

12.5.1. Build and Deploy the CarMart Quickstart to the JBoss Server

The following procedure provides directions to build and deploy the CarMart application to a JBoss server.



NOTE

The directions for this quickstart are for use with the selected JBoss server, not 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, refer to [Chapter 4, *Install and Use the Maven Repositories*](#)
3. Select a JBoss server to use (JBoss Enterprise Application Platform 6 (or better) or JBoss Application Server 7 (or better)).

Procedure 12.1. Build and Deploy CarMart to the JBoss Server

1. Start the JBoss Server

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
```

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-as-carmart.war**) is deployed to the running instance of the selected JBoss server.

[Report a bug](#)

12.5.2. View the CarMart Quickstart on the JBoss Application Server

The following procedure outlines how to view the CarMart quickstart on the JBoss Application Server:

Prerequisite

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

Procedure 12.2. View the CarMart Quickstart on the JBoss Application Server

- 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-as-carmart
```

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

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

[Report a bug](#)

12.5.3. Remove the CarMart Quickstart from the JBoss Server

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

Procedure 12.3. Remove an Application from the JBoss Server

- To remove an application, use the following command:

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

[Report a bug](#)

12.6. THE CARMART QUICKSTART TO A SERVER

12.6.1. 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. 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, refer to [Chapter 4, Install and Use the Maven Repositories](#)
3. Select either JBoss Enterprise Web Server 2 (or better) or Tomcat 7 (or better) for your application and install it.

Procedure 12.4. 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:

```
$TOMCAT7_HOME/bin/catalina.sh run
```

For Windows users:

```
$TOMCAT7_HOME\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-as-carmart.war**) is deployed to the running instance of the selected server.

[Report a bug](#)

12.6.2. 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 12.5. View the CarMart Quickstart

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

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

[Report a bug](#)

12.6.3. Remove the CarMart Quickstart from the Server

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

Procedure 12.6. Remove an Application from the Server

- To remove an application, use the following command:

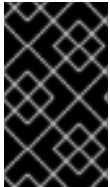
```
$ mvn tomcat:undeploy -Plibrary-tomcat
```

[Report a bug](#)

12.7. THE CARMART QUICKSTART IN REMOTE CLIENT-SERVER MODE

12.7.1. Build and Deploy 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.



IMPORTANT

This quickstart deploys to JBoss Enterprise Application Platform or JBoss Application Server. The application can not 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, refer to [Chapter 4, Install and Use the Maven Repositories](#)
3. Select a JBoss server to use (JBoss Enterprise Application Platform 6 (or better) or JBoss Application Server 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 12.7. 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"
  indexing="NONE">
<locking isolation="REPEATABLE_READ"
  striping="false"
  acquire-timeout="20000"
  concurrency-level="500"/>
```

```
<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-quickstarts-6.1/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](#)

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

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

Prerequisite

The CarMart quickstart must be built and deployed before viewed.

Procedure 12.8. 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-as-carmart
```

[Report a bug](#)

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

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

Procedure 12.9. 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 13. THE FOOTBALL QUICKSTART ENDPOINT EXAMPLES

13.1. 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-6.1/rest-endpoint`
- `jboss-datagrid-quickstarts-6.1/hotrod-endpoint`
- `jboss-datagrid-quickstarts-6.1/memcached-endpoint`

[Report a bug](#)

13.2. 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 JBoss Application Server 7.x.
- Maven 3.0 or better.
- Configure the Maven Repository. For details, refer to [Chapter 4, Install and Use the Maven Repositories](#)

[Report a bug](#)

13.3. 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.



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, refer to [Chapter 4, Install and Use the Maven Repositories](#)

Procedure 13.1. 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:5.2"
  default-cache-container="local">

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

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

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

      <transaction mode="NONE"/>

    </local-cache>

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

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

      <transaction mode="NONE"/>

    </local-cache>

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

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

    <local-cache name="teams"
      start="EAGER"
      batching="false"
      indexing="NONE" >

      <!-- Define the locking
      isolation of this cache -->

      <locking isolation="REPEATABLE_READ"
        acquire-timeout="20000"
        concurrency-level="500"

```

```

        striping="false" />

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

        <!-- Define the JdbcBinaryCacheStores
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, refer to 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 JBOSS DATA GRID

CHAPTER 14. REMOVE JBOSS DATA GRID

14.1. 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 14.1. 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](#)

14.2. 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 14.2. 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

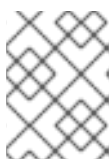
This example will cover the steps to install the JBoss Enterprise Application Platform 6 Maven Repository using Sonatype Nexus Maven Repository Manager. For more complete instructions, see [Sonatype Nexus: Manage Artifacts](#).

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...** dropdown, 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 you want the artifact to be available in a repository group. Do not continue with this procedure if this is not what you want.
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

Note that 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>jboss-eap-repository</id>
          <name>JBoss EAP Maven Repository</name>
          <url>file:///path/to/repo/jboss-eap-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-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>jboss-eap-repository</id>
          <name>JBoss EAP Maven Repository</name>
          <url>file:///path/to/repo/jboss-eap-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-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 6.1.0-4.400 Rebuild with publican 4.0.0	2013-10-31	Rüdiger Landmann
Revision 6.1.0-4 Updated with new product name.	Tue Aug 06 2013	Misha Husnain Ali
Revision 6.1.0-3 Patch for BZ-948988.	Sun Apr 07 2013	Misha Husnain Ali