Red Hat JBoss Data Grid 7.2

Feature Support Document

For use with Red Hat JBoss Data Grid 7.2

Last Updated: 2018-05-16

Red Hat JBoss Data Grid 7.2 Feature Support Document

For use with Red Hat JBoss Data Grid 7.2

Legal Notice

Abstract

Details JBoss Data Grid feature support and technology preview status information.

Table of Contents

CHAPTER 1. INTRODUCTION

1.1. INTRODUCTION

This document clarifies some important information related to features and support for Red Hat JBoss Data Grid, such as:

- The two JBoss Data Grid Usage Modes
- Supported JBoss Data Grid features
- JBoss Data Grid features that are limited to a technology preview.

1.2. ABOUT USAGE MODES

Red Hat JBoss Data Grid offers two usage modes:

- Remote Client-Server mode
- Library mode

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

Library mode 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 Library mode includes JBoss Enterprise Web Server and JBoss Enterprise Application Platform (see https://access.redhat.com/articles/2435931 for details about supported containers). Additionally, Library mode is supported outside the listed containers as a standalone application.

1.3. FEATURES AND USAGE MODES

The following table presents a list of features and indicates the usage mode for each feature. Red Hat JBoss Data Grid 7.2 includes full support for both Remote Client-Server mode and Library mode.

Table 1.1. JBoss Data Grid Features

Feature	Remote Client-Server Mode (Supported)	Library Mode (Supported)
File Cache Store and Loading	✓	✓
JDBC Cache Store and Loading	✓	✓
LevelDB Cache Store and Loading	✓	✓
Cassandra Cache Store and Loading	✓	✓

Feature	Remote Client-Server Mode (Supported)	Library Mode (Supported)
Cache Passivation	✓	✓
Remote Cache Store	✓	✓
Cluster Cache Store	✓	✓
Asynchronous Store	✓	✓
Cluster Configuration Using UDP	✓	✓
Cluster Configuration Using TCP	✓	✓
Mortal and Immortal Data	✓	✓
Eviction Strategy	✓	✓
Expiration	✓	✓
Unscheduled Write-behind Cache Store	✓	✓
Write-through Cache Store	✓	✓
Clustering Mode (local)	✓	✓
Clustering Mode (replicated)	✓	✓
Clustering Mode (invalidation)	✓	✓
Clustering Mode (distribution)	✓	✓

Feature	Remote Client-Server Mode (Supported)	Library Mode (Supported)
Asynchronous Clustering Modes	✓	✓
Marshalling	✓	✓
Management Using JMX	✓	✓
Cross-Datacenter Replication and State Transfer	✓	✓
JBoss Operations Network (JON) Integration and Plugin	✓	✓
Asymmetric Cluster	✓	✓
Command Line Interface (CLI)	✓	✓
Role-based Access Control	✓	✓
Node Authentication and Authorization	✓	✓
Encrypted Communication Within the Cluster	✓	✓
Per Invocation Flags	✓	✓
Customizable Network Partition Handling	✓	✓
Spring Integration	✓	✓
Apache Camel Component for JBoss Fuse	✓	✓
Querying (by values)	✓	✓

Feature	Remote Client-Server Mode (Supported)	Library Mode (Supported)
Continuous Queries	✓	✓
Clustered Listeners and Notifications for Cache Events	✓	✓
Near Caching	✓	✓
JSR-107 Support	✓	✓
CDI	✓	✓
Asynchronous API	✓	✓
Distributed Streams[1]	✓	✓
Off Heap Cache Storage	✓	✓
Ickle Query Language	✓	✓
EAP Integration	✓	✓
Deploy custom cache store to JDG Server	✓	
Connection Pooling with JDBC Cache Stores	✓	
REST Interface	✓	
Memcached Interface	✓	
Hot Rod Java client	✓	

Feature	Remote Client-Server Mode (Supported)	Library Mode (Supported)
Hot Rod C++ Client	✓	
Hot Rod .NET Client	✓	
Hot Rod Node.js Client	✓	
Data Compatibility Between Client-server Protocols	✓	
Data Compatibility Between Hot Rod Java and C++ Client	✓	
Rolling Upgrades for Hot Rod Cluster	✓	
Controlled Shutdown and Restart of Cluster	✓	
JBoss Data Grid's Hot Rod Client as a JBoss EAP Module	✓	
Externalizing HTTP sessions from JBoss EAP 7 to remote JDG cluster	✓	
Externalizing HTTP sessions from JBoss Web Server to remote JDG cluster	✓	
Remote Task Execution	✓	
Apache Spark with Scala 2.11	✓	
Apache Hadoop Integration	✓	
Administration Console	✓	

Feature	Remote Client-Server Mode (Supported)	Library Mode (Supported)
READ_COMMITTED and REPEATABLE_READ Isolation Modes		✓
Lazy Deserialization		✓
Using the infinispan.xml File in Conjunction with APIs		✓
Grouping API		✓
Java Transactional API (JTA) Support and Configuration		✓
Java Transactional API (JTA) Deadlock Detection		✓
Transaction Recovery		✓
Transaction and Batching		✓
Key Affinity		✓
Distributed Execution Framework		✓
JPA Cache Store		✓
JBoss Data Grid as a JBoss EAP Module		✓
JDG as Lucene Directory		✓

1: Distributed Streams are available in JBoss Data Grid's Remote Client-Server Mode via Remote Task Execution.

1.4. HOT ROD FEATURES BY LANGUAGE

The following table presents a list of Hot Rod features and indicates the languages supported for each feature.

Table 1.2. Hot Rod Client Features by Language

	Java	C++	C#	Node.js
Cross-site Failover	✓	✓	✓	✓
Authentication	✓	✓	✓	✓
TLS-based Encryption	✓	✓	✓	✓
Server Name Indication (SNI)	✓	✓	✓	✓
Asynchronous API	✓	✓	✓	
Near Caching	✓	✓	✓	
Continuous Queries	✓	✓	✓	
Remote Event Listeners	✓			✓
Remote Querying	✓			
Remote Execution	✓			

1.5. TECHNOLOGY PREVIEW FEATURES

The following features are included in Red Hat JBoss Data Grid 7.2 as a technology preview only:

- Querying data via the REST interface: This technology preview feature provides the ability to query data via the REST interface using Ickle queries in JSON format. For more information, see Querying Data via the REST Interface in the Developer Guide.
- The Hot Rod C++ Client includes the following features as technology preview:
  - Remote Querying
  - Remote Execution

- The Hot Rod C# Client includes the following features as technology preview:
  - Remote Querying
  - Remote Execution

WARNING

Technology Preview features are not supported with Red Hat production service level agreements (SLAs), may not be functionally complete, and are not recommended to be used for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.