Red Hat JBoss Data Virtualization 6.2

Migration Guide

This guide is for installation teams

Last Updated: 2017-10-20

Red Hat JBoss Data Virtualization 6.2 Migration Guide

This guide is for installation teams

Red Hat Customer Content Services

Legal Notice

Abstract

This document guides users through the process for migrating to Red Hat JBoss Data Virtualization 6 from previous versions.

Table of Contents

CHAPTER 1. READ ME **3**
1.1. BACK UP YOUR DATA 3
1.2. VARIABLE NAME: EAP_HOME 3
1.3. VARIABLE NAME: MODE 3
1.4. RED HAT DOCUMENTATION SITE 3

CHAPTER 2. MIGRATION **4**
2.1. PRODUCT NAME CHANGE 4
2.2. MIGRATING JBOSS DATA VIRTUALIZATION 4
2.3. MIGRATING HIERARCHICAL DATA 5
2.4. LDAP COMPLIANCE 6

CHAPTER 3. CHANGES **7**
3.1. CHANGES TO JBOSS DATA VIRTUALIZATION 7
3.2. CHANGES TO THE HIERARCHICAL DATABASE SYSTEM 7

CHAPTER 4. BACKUP AND RESTORE **11**
4.1. MIGRATING FROM A PREVIOUS RELEASE 11
4.2. THE REPOSITORY MANAGER 11
4.3. BACKUP A REPOSITORY 12
4.4. RESTORE A REPOSITORY 12

CHAPTER 5. SUPPORTED CONFIGURATIONS **14**
5.1. SUPPORTED DATA SOURCES AND TRANSLATORS 14

APPENDIX A. REVISION HISTORY **17**

CHAPTER 1. READ ME

1.1. BACK UP YOUR DATA

WARNING

Red Hat recommends that you back up your system settings and data before undertaking any of the configuration tasks mentioned in this book.

1.2. VARIABLE NAME: EAP_HOME

EAP_HOME refers to the root directory of the Red Hat JBoss Enterprise Application Platform installation on which JBoss Data Virtualization has been deployed.

1.3. VARIABLE NAME: MODE

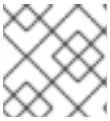
MODE will either be `standalone` or `domain` depending on whether JBoss Data Virtualization is running in standalone or domain mode. Substitute one of these whenever you see **MODE** in a file path in this documentation. (You need to set this variable yourself, based on where the product has been installed in your directory structure.)

1.4. RED HAT DOCUMENTATION SITE

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

CHAPTER 2. MIGRATION

2.1. PRODUCT NAME CHANGE



NOTE

This product was previously called JBoss Enterprise Data Services, or EDS.

2.2. MIGRATING JBOSS DATA VIRTUALIZATION

This topic lists tasks that you might have to implement in order to migrate to JBoss Data Virtualization 6.

What will I have to do?

- To migrate to Red Hat JBoss Data Virtualization 6, users will need to install a fresh instance to redeploy customer applications and artifacts.
- If users are migrating from a version prior to JBoss Enterprise Data Services 5.3.1, then the REST WARs will need to be regenerated due to changes in CXF. Otherwise the WAR will not deploy due to the WSDL not being found.
- Previously, null values were ranked low by default. This could have a negative impact on performance and is not needed in many situations. The ranking of null values are no longer pushed down by default. Users will have to set the org.teiid.pushdownDefaultNullOrder property to preserve the old behavior if required.

NOTE

- The old update procedure syntax is no longer supported. Therefore, users may have to revalidate models and rebuild the VDB.
- Users may have to reimport VDBs stored by the hierarchical database in order to update node types.
- Any custom translator or connector (resource adapter) will require migrating. It will need to be recompiled and new configuration and packaging will be needed. This is due to the changes from EAP 5 to EAP 6. Also, the translator documentation should be read, because additional features have been added (for example, DDL support), that would require additional updates to the translator.

How will migration impact modeling?

- Users may have issues when validating transformations after importing a project set that were built in the previous version of Teiid Designer. Users may have to adjust transformations in their models to account for this.

NOTE

When you migrate models containing REST procedures from 7.7.x version, edit the XMI file and change the key names. The designer tool then displays the value for the given properties.

```
key= rest:uri is now key= REST:URI
And
key= rest:restMethod is now key= REST:METHOD
```

- When importing any of the EDS project, import all the `.xmi` files one-by-one. This helps you to eliminate all the possible validation errors when importing the complete project set. However, do not import the `.project` file.

WARNING

Do not remove old data before verifying the migration to JBoss Data Virtualization 6 was successful.

2.3. MIGRATING HIERARCHICAL DATA

Data stored in the provided hierarchical database can be migrated from previous versions of JBoss Data Virtualization using the new backup and restore functions.

To migrate a repository, back it up from your previous installation, then restore it in your new JBoss Data Virtualization 6 installation.

Each backup contains all content in all workspaces for a single repository. Therefore, to migrate to release 6, the backup/restore process will need to be followed for each repository.

NOTE

A migration tool will be provided to backup from previous installations, compatible with the new backup and restore functions.

WARNING

Do not remove old data before verifying the migration to JBoss Data Virtualization 6 is successful.

See Also:

- Section 4.1, “Migrating from a Previous Release”

2.4. LDAP COMPLIANCE

It is recommended that customers who have utilized the internal JDBC membership domain from releases prior to MetaMatrix 5.5 migrate those users and groups to an LDAP compliant directory server.

Refer to the *Red Hat JBoss Enterprise Application Platform Security Guide* for directions on using an LDAP directory server. Please contact technical support if you require additional guidance in the migration process.

Several free and open source directory servers include:

- The Fedora Directory Server - http://directory.fedoraproject.org/
- Open LDAP - http://www.openldap.org/
- Apache Directory Server - http://directory.apache.org/

CHAPTER 3. CHANGES

3.1. CHANGES TO JBOSS DATA VIRTUALIZATION

Installation

- In previous releases, JBoss Data Virtualization depended on the Red Hat JBoss Services Oriented Architecture Platform (SOA-P). In this release, JBoss Data Virtualization is installed on the Red Hat JBoss Enterprise Application Platform (EAP) directly. Users will be able to install on top of an existing JBoss EAP installation or with a fresh installation.

- In previous releases, JBoss Data Virtualization could operate under a number of *profiles*. In this release, JBoss Data Virtualization can be installed in one of the two JBoss EAP *modes*: standalone mode (default) or domain mode (for clustering).

Quickstarts

Quickstarts now rely on Maven instead of Apache Ant.

Tooling Support

Teiid Designer and ModeShape Tools plugins for Red Hat JBoss Developer Studio will support both version 5 and 6 of JBoss Data Virtualization.

3.2. CHANGES TO THE HIERARCHICAL DATABASE SYSTEM

APIs

For both releases 5 and 6, client applications use the standard JCR 2.0 API to interact with repositories. So most (if not all) of your application code can remain unchanged.

A small API (`org.modeshape.jcr.api`) was provided in release 5 that extended the standard JCR API with a few additional capabilities. This release supports and slightly expands on this API. Several of the methods and interfaces in the API were deprecated in the previous release. These have now been removed.

The URLs for the RESTful API have changed. This release provides an improved RESTful API which is now the default. The RESTful client library is compatible with the database system provided in both versions 5 and 6.

The WebDAV API has been improved and the JDBC driver is also still supported.

Caching

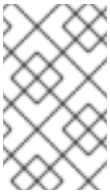
In release 5, a single Service Provider Interface (SPI) was used for both storage and access connectors. However, this has been changed as part of a move to a more scalable caching solution.

In this release, each repository uses a single cache for both its caching and storage system. All workspace content (except for binary value storage) and system content is stored within this single cache.

Federation

In this release, the JCR API can be used to access information in external systems. The concept of a *connector* has been reintroduced as a mechanism to achieve this. Repositories will no longer consist entirely of federated content. Instead, every repository will store its own content, but you will be

able to federate and integrate content from external systems into the repository.



NOTE

Technically, even in the previous release a storage connector was required to store the repository's system content, so it was never actually possible to have a repository that consisted entirely of federated content.

Binary Storage

Repositories can now handle extremely large binary values with several storage options:

- Binary values can be stored on the local file system. This can be a regular directory, a network share or even a temporary directory. This option is generally fast, safe (as safe as your file system), and native locks are used to prevent multiple processes from conflicting. It is recommended for local (non-clustered), embedded repositories.
- Binary values can be stored in cache. Although it is possible to use the same cache as the rest of the repository, it is often better to use two other caches and to configure those caches specifically for what they store. For example, one cache is used to store metadata about the binary values; this metadata is small, lightweight, and can thus be replicated across the cluster. The other cache is used to store the actual binary data, separated into chunks (usually up to 1MB in size), and for this type of cache, distributing the data across the cluster is often desirable.
- Binary values can be stored in a relational database. This is recommended only when you intend to persist all content inside a relational database. The binary data is broken into chunks (usually up to 1MB in size).
- Binary values can be stored in MongoDB. This storage option has not been thoroughly tested, but can be considered as an option.
- Binary values can be stored in a custom store. A Service Provider Interface (SPI) is now provided for implementing your own binary storage.

Sequencers

Each repository is now configured with its own sequencers. Implementing custom sequencers is simpler since sequencers generate additional content by using the JCR API directly, rather than the proprietary graph API used in the previous release. Sequencer implementations are also able to register the node types programmatically, which simplifies the overall configuration for a repository.

MIME Type Detection

There are two additional methods to the org.modeshape.jcr.api.Binary interface to obtain the MIME type. Your applications can use these methods to discover the MIME type for a binary value and, for example, to set the "jcr:mimeType" property on the node.

The Service Provider Interface (SPI) for custom MIME type detectors has been removed. Instead, the Apache Tika framework is used, which has several MIME type detectors and provides its own SPI for custom detectors.

Text Extractors

A simple Service Provider Interface (SPI) has been added for simple creation of custom extractors.

Configuration

In this release, there is one JSON (previously XML) configuration file for each repository, each "deployed" to an engine. As a result, it is now possible to deploy and undeploy repositories dynamically even when the engine is running and other repositories are in use.

A JSON schema dictates the allowed structure of the configuration files. This means they can be validated:

```
Problems problems = config1.validate();
if ( problems.hasProblems() ) {
    // Output a summary of the problems (with line numbers) ...
    System.out.println(problems);
}
```

Sensible defaults are in place for settings that are not configured explicitly. The following is now a valid configuration:

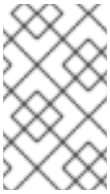
```
{ "name" : "my-repo" }
```

You can now load configuration via the following means:

- read from a java.io.File
- read from a resolved java.net.URL
- read from a String containing a URL or a path to a file on the file system or classpath
- read a string containing the configuration

It is also possible to access the configuration of a running repository, change the configuration, and then update the running repository:

```
// Get the configuration ...
RepositoryConfiguration config = repository1.getConfiguration();

// Edit the configuration (which is a JSON document) to change a value
...
Editor editor = config.edit();
editor.getOrCreateDocument(FieldName.STORAGE)
      .getOrCreateDocument(FieldName.BINARY_STORAGE);
      .setNumber(FieldName.MINIMUM_BINARY_SIZE_IN_BYTES,
newLargeValueSizeInBytes);
Changes changes = editor.getChanges();

// Apply the changes to the deployed repository ...
Future<Boolean> future = engine.update(config.getName(), changes);

// And optionally wait until the repository configuration is updated ...
future.get();
```



NOTE

Many configuration changes can now be applied to a repository while it is running, but not everything. For example, changing *where* data is stored will apply only after the repository is restarted.

Migrating Content

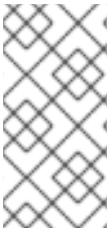
The database system now provides efficient backup and restore functionality that works at the repository level. Each backup contains all of the content in all workspaces for a single repository. Backups can be used to restore a repository to an earlier state, and it also serves as a mechanism for migrating from the previous release.

CHAPTER 4. BACKUP AND RESTORE

4.1. MIGRATING FROM A PREVIOUS RELEASE

Backup and restore can be used to migrate content stored in the provided hierarchical database system.

This is the proposed way for users to migrate such data for release 6.



NOTE

When migrating from a version of JBoss Data Virtualization prior to version 6.0.0, a migration tool will be provided to backup the repository from the previous installation. The backup can be restored using the **`restoreRepository`** method on the new (and empty) repository.

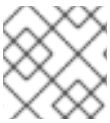
4.2. THE REPOSITORY MANAGER

The org.modeshape.jcr.api.RepositoryManager interface contains the backup and restore functions.

```
public interface RepositoryManager {

    ...

    Problems backupRepository( File backupDirectory ) throws
RepositoryException;

    ...

    Problems restoreRepository( File backupDirectory ) throws
RepositoryException;

    ...

}
```

The following code demonstrates how to access the repository manager from a standard authenticated JCR session:

```
javax.jcr.Repository repository = ...
javax.jcr.Credentials credentials = ...
String workspaceName = ...
javax.jcr.Session session = repository.login(credentials,workspaceName);
org.modeshape.jcr.api.Session msSession =
(org.modeshape.jcr.api.Session)session;
org.modeshape.jcr.api.RepositoryManager repoMgr =
((org.modeshape.jcr.api.Session)session).getWorkspace().getRepositoryManag
er();
```



NOTE

Which workspace is used by the session is not important.

4.3. BACKUP A REPOSITORY

The **backupRepository** method provided by the org.modeshape.jcr.api.RepositoryManager interface is used to create a backup of the entire repository, including all workspaces that existed when the backup was initiated.

This method blocks until the backup is completed, so it is the caller's responsibility to invoke the method asynchronously if that is desired.

When this method is called on a repository that is being actively used, all of the changes made while the backup process is underway will be included. At some point near the end of the backup process, however, additional changes will be excluded from the backup. This means that each backup contains a fully-consistent snapshot of the entire repository as it existed near the time at which the backup completed.

The following code demonstrates usage of the backup method:

```
org.modeshape.jcr.api.RepositoryManager repoMgr = ...
java.io.File backupDirectory = ...
Problems problems = repoMgr.backupRepository(backupDirectory);
if ( problems.hasProblems() ) {
    System.out.println("Problems restoring the repository:");
    // Report the problems (we'll just print them out) ...
    for ( Problem problem : problems ) {
       System.out.println(problem);
    }
} else {
    System.out.println("The backup was successful");
}
```

Each backup is stored on the file system in a directory that contains a series of GZIP-ed files (each containing representations of approximately 100K nodes) and a subdirectory in which all the large BINARY values are stored.

NOTE

It is the application's responsibility to initiate each backup operation. There currently is no way to configure a scheduled backup. Doing so would add significant complexity.

4.4. RESTORE A REPOSITORY

Once you have a complete backup on disk, you can then restore a repository back to the state captured within the backup using the **restoreRepository** method provided by the org.modeshape.jcr.api.RepositoryManager interface.

To do this, start a repository (or perhaps a new instance of a repository with a different configuration) and, before it is used by any applications, restore the content.

The following code demonstrates usage of the restore method:

```
org.modeshape.jcr.api.RepositoryManager repoMgr = ...
java.io.File backupDirectory = ...
Problems problems = repoMgr.restoreRepository(backupDirectory);
if ( problems.hasProblems() ) {
    System.out.println("Problems backing up the repository:");
```

```
    // Report the problems (we'll just print them out) ...
    for ( Problem problem : problems ) {
         System.out.println(problem);
    }
} else {
    System.out.println("The restoration was successful");
}
```

Once a restore succeeds, the newly-restored repository will be restarted and ready for use.

CHAPTER 5. SUPPORTED CONFIGURATIONS

5.1. SUPPORTED DATA SOURCES AND TRANSLATORS

The following table provides a list of data sources and translators that are supported by Red Hat.

Table 5.1. Supported Data Sources and Translators

Data Source	Translator	Supported DV Version	Driver
Apache Hive 12	-	6.0+	-
Apache Solr	solr	6.1+	-
Cloudera Hadoop	-	6.1+	-
EDS 5.x	teiid	6.0+	-
Files – delimited, fixed length	file	6.0+	-
Generic Datasource-JDBC ansi	jdbc-ansi	6.0+	-
Generic Datasource-JDBC simple	jdbc-simple	6.0+	-
Google Spreadsheet	-	6.0+	-
Greenplum 4.x	postgresql	6.0+	-
Hortonworks Hadoop	-	6.1.+	-
HSQL (for test/examples only)	-	-	-
IBM DB2 10	db2	6.1+	Universal Driver v4.x
IBM DB2 9.7	db2	6.0+	Universal Driver v4.x
Ingres 10	ingres	6.0+	-
Intel Hadoop	-	6.1+	-
JBoss Data Grid 6.4 (remote client - hotrod)	infinispan-cache-dsl	6.2+	-
JBoss Data Grid 6.4 (library mode)	infinispan-cache	6.0 - post GA, 6.1+	-
LDAP/ActiveDirectory v3	ldap	6.0+	-
Mainframe (CICS,IMS,VSAM)	-	6.0+	-

Data Source	Translator	Supported DV Version	Driver
MariaDB	mysql5	6.1+	-
ModeShape/JCR 3.1	-	6.0+	-
MongoDB 2.2	mongodb	6.0	post GA, 6.1+ -
MS Access 2010	-	6.0+	-
MS Access 2013	-	6.0+	-
MS Excel 2010	excel	6.0+	-
MS Excel 2013	excel	6.0+	-
MS SQL Server 2008	sqlserver	6.0+	Microsoft SQL Server JDBC Driver 4
MS SQL Server 2012	sqlserver	6.0+	Microsoft SQL Server JDBC Driver 4
MySQL 5.1	mysql5	6.0+	V5.1
MySQL 5.5	mysql5	6.0+	V5.5
Netezza 6.0.2	netezza	6.0+	-
Oracle 10g R2	oracle	6.0+	Oracle JDBC Driver v10
Oracle 11g RAC	oracle	6.0+	Oracle JDBC Driver v11
Oracle 12c	oracle	6.0 - post GA, 6.1+	Oracle JDBC Driver v12
PostgreSQL 8.4	postgresql	6.0+	-
PostgreSQL 9.2	postgresql	6.0+	-
REST/JSON over HTTP	ws	-	-
RHEL 5.5/6 PostgreSQL config	-	6.0+	-

Data Source	Translator	Supported DV Version	Driver
Salesforce.com API 22	salesforce	6.0+	-
SAP Netweaver Gateway (OData)	sap-nw-gateway	6.1+	-
Support SAP Service Registry as a Data Source	-	6.2+	-
Sybase ASE 15	sybase	6.0+	jConnect JDBC3.0 v7
Teradata Express 12	teradata	6.0+	-
Webservices	ws	6.0+	-
XML Files	FILE	6.0+	-

NOTE

MS Excel is supported in so much as there is a write procedure.

APPENDIX A. REVISION HISTORY

Revision 6.2.0-03 **Tue Aug 4 2015** **David Le Sage**

Updates for 6.2.