Abstract

This guide provides information about how to migrate your application from previous versions of Red Hat JBoss Enterprise Application Platform.
Table of Contents

CHAPTER 1. INTRODUCTION  ................................................................................................. 7
 1.1. ABOUT RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 7 7
 1.2. ABOUT THE MIGRATION GUIDE 7
 1.3. ABOUT MIGRATIONS AND UPGRADES 7
 1.4. ABOUT THE USE OF EAP_HOME IN THIS DOCUMENT 8

CHAPTER 2. PREPARE FOR MIGRATION  .............................................................................. 9
 2.1. PREPARATION OVERVIEW 9
 2.2. REVIEW THE JAVA EE 7 FEATURES 9
 2.3. REVIEW WHAT’S NEW IN JBOSS EAP 7 9
 2.4. REVIEW THE LIST OF DEPRECATED AND UNSUPPORTED FEATURES 10
 2.5. REVIEW THE JBOSS EAP GETTING STARTED MATERIAL 11
 2.6. MIGRATION ANALYSIS AND PLANNING 11
 2.7. BACK UP IMPORTANT DATA AND REVIEW SERVER STATE 12
 2.8. MIGRATING AN RPM INSTALLATION 13
 2.9. MIGRATE JBOSS EAP RUNNING AS A SERVICE 13

CHAPTER 3. TOOLS TO ASSIST IN MIGRATION  .................................................................. 14
 3.1. USE RHAMT TO ANALYZE APPLICATIONS FOR MIGRATION 14
 3.2. USE THE JBOSS SERVER MIGRATION TOOL TO MIGRATE SERVER CONFIGURATIONS 14

CHAPTER 4. SERVER CONFIGURATION CHANGES  ................................................................. 15
 4.1. SERVER CONFIGURATION MIGRATION OPTIONS 15
    JBoss Server Migration Tool 15
    Management CLI Migrate Operation 15
 4.2. MANAGEMENT CLI MIGRATION OPERATION 15
    Start the Server and the Management CLI 16
    Migrate the JacORB, Messaging, and Web Subsystems 17
 4.3. LOGGING MESSAGE PREFIX CHANGES 20
 4.4. WEB SERVER CONFIGURATION CHANGES 21
    4.4.1. Replace the Web Subsystem with Undertow 21
    4.4.2. Migrate JBoss Web Rewrite Conditions 22
    4.4.3. Migrate JBoss Web System Properties 24
    4.4.4. Migrate JBoss Web jboss-web.xml Overlay 25
    4.4.5. Migrate Global Valves 25
        Migrate JBoss Web Valves 25
        JDBCAccessLogValve Manual Migration Procedure 26
 4.5. JGROUPS SERVER CONFIGURATION CHANGES 27
    4.5.1. JGroups Defaults to a Private Network Interface 27
    4.5.2. JGroups Channels Changes 27
 4.6. INFINISPAN SERVER CONFIGURATION CHANGES 28
    4.6.1. Infinispan Default Cache Configuration Changes 28
    4.6.2. Infinispan Cache Strategy Changes 28
    4.6.3. Configuring Custom Stateful Session Bean Cache for Passivation 28
 4.7. EJB SERVER CONFIGURATION CHANGES 29
    DuplicateServiceException 29
 4.8. MESSAGING SERVER CONFIGURATION CHANGES 29
    4.8.1. Messaging Subsystem Server Configuration Changes 30
        Management Model 30
        Messaging Subsystem Migration and Forward Compatibility 30
        Change in Behavior of forward-when-no-consumers Attribute 31
        Change in Default Cluster Load Balancing Policy 31
Messaging Subsystem XML Configuration

4.8.2. Migrate Messaging Data
4.8.2.1. Migrate Messaging Data Using Export and Import
   Export Messaging Data from the Previous Release
   Import the XML Formatted Messaging Data
   Recovering from an Import Messaging Data Failure
4.8.2.2. Migrate Messaging Data Using a JMS Bridge
   Configure the JBoss EAP 6 Server
   Configure the JBoss EAP 7 Server
   Migrate the Data
4.8.2.3. Mapping Messaging Folder Names
4.8.2.4. Backing Up Messaging Folder Data
4.8.3. Migrate JMS Destinations
4.8.4. Migrate Messaging Interceptors
4.8.5. Replace Netty Servlet Configuration
4.8.6. Changes in JMS Serialization Behavior Between Releases

4.9. JMX MANAGEMENT CHANGES

4.10. ORB SERVER CONFIGURATION CHANGES

4.11. MIGRATE THE THREADS SUBSYSTEM CONFIGURATION

4.12. MIGRATE THE REMOTING SUBSYSTEM CONFIGURATION

4.13. WEBSOCKET SERVER CONFIGURATION CHANGES

4.14. SINGLE SIGN-ON SERVER CHANGES

4.15. Datasource Configuration Changes
4.15.1. JDBC Datasource Driver Name
   Driver Containing a Single Class
   Driver Containing Multiple Classes

4.16. SECURITY SERVER CONFIGURATION CHANGES

4.17. TRANSACTIONS SUBSYSTEM CHANGES
   Removed Transactions Subsystem Attributes
   Deprecated Transactions Subsystem Attributes

4.18. CHANGES TO MOD_CLUSTER CONFIGURATION

CHAPTER 5. APPLICATION MIGRATION CHANGES

5.1. WEB SERVICES APPLICATION CHANGES
5.1.1. JAX-RPC Support Changes
5.1.2. Apache CXF Spring Web Services Changes
   Apache CXF Interceptors
   Apache CXF Features
   Apache CXF HTTP Transport
5.1.3. WS-Security Changes
5.1.4. JBoss Modules Structure Change
5.1.5. Bouncy Castle Requirements and Changes
5.1.6. Apache CXF Bus Selection Strategy
5.1.7. JAX-WS 2.2 Requirements for WebServiceRef
5.1.8. IgnoreHttpsHost CN Check Change
5.1.9. Server Side Configuration and Class Loading
5.1.10. Deprecation of Java Endorsed Standards Override Mechanism
5.1.11. Specification of Descriptor in EAR Archive

5.2. UPDATE THE REMOTE URL CONNECTOR AND PORT

5.3. MESSAGING APPLICATION CHANGES
5.3.1. Replace or Update JMS Deployment Descriptors
5.3.2. Update External JMS Clients
5.3.3. Replace the HornetQ API
5.4. JAX-RS AND RESTEASY APPLICATION CHANGES

5.4.1. RESTEasy Deprecated Classes
   Interceptor and MessageBody Classes
   Client API
   StringConverter

5.4.2. Removed or Protected RESTEasy Classes
   ResteasyProviderFactory Add methods
   Additional Classes Removed From RESTEasy 3

5.4.3. Additional RESTEasy Changes
   SignedInput and SignedOuput
   Security Filters
   Client-side Filters
   Asynchronous HTTP Support
   Server-side Cache
   YAML Provider Setting Changes
   SerializableProvider

5.4.4. RESTEasy SPI Changes
   SPI Exceptions
   InjectorFactory and Registry

5.4.5. Jackson Provider Changes

5.4.6. Spring RESTEasy Integration Changes

5.4.7. RESTEasy Jettison JSON Provider Changes

5.5. CDI 1.2 APPLICATION CHANGES
   Bean Archives
   Clarification of Conversation Resolution
   Observer Resolution

5.6. MIGRATE EXPLICIT MODULE DEPENDENCIES
   Review Dependencies for Availability
   Dependencies That Require Annotation Scanning

5.7. HIBERNATE AND JPA MIGRATION CHANGES
   5.7.1. Hibernate ORM 3.0
   5.7.2. Hibernate ORM 4.0 - 4.3
   5.7.3. Hibernate ORM 5
   5.7.4. Migrating to Hibernate ORM 5
      Removed and Deprecated Classes
      Other Changes to Classes and Packages
      Type Handling
      Transaction Management
      Other Hibernate ORM 5 Changes

5.8. HIBERNATE SEARCH CHANGES
   Hibernate Search Mapping Changes
      Indexing of id Fields of Embedded Relations
      Number and Date Index Formatting Changes
   Miscellaneous Hibernate Search Changes
   Hibernate Search Renamed and Repackaged Classes
   Lucene - Renamed and Repackaged Classes
   Hibernate Search Deprecated APIs
      Hibernate Search Deprecated Interfaces
      Hibernate Search Deprecated Classes
      Hibernate Search Deprecated Enums
      Hibernate Search Deprecated Annotations
      Hibernate Search Deprecated Methods
      Hibernate Search Deprecated Constructors
Changes Impacting Advanced Integrators
5.9. MIGRATE ENTITY BEANS TO JPA 72
5.10. JPA PERSISTENCE PROPERTY CHANGES 74
5.11. MIGRATE EJB CLIENT CODE 74
  5.11.1. Update the Default Remote Connection Port 74
  5.11.2. Update the Default Connector 75
  5.11.3. Migrate Remote Naming Client Code 75
5.12. MIGRATE DEPLOYMENT PLAN CONFIGURATIONS 75
5.13. MIGRATE CUSTOM APPLICATION VALVES 76
  Migrate Valves Configured in Deployments 76
  Migrate Custom Authenticator Valves 76
5.14. SECURITY APPLICATION CHANGES 77
  5.14.1. Migrate Authenticator Valves 77
  5.14.2. PicketLink Changes 77
  5.14.3. Other Security Application Changes 77
5.15. JBOSS LOGGING CHANGES 77
5.16. JAVASERVER FACES (JSF) CODE CHANGES 78
  Dropped Support for JSF 1.2 78
  Compatibility Issue Between JSF 2.1 and JSF 2.2 78
5.17. MODULE CLASS LOADING CHANGES 78
5.18. APPLICATION CLUSTERING CHANGES 79
  5.18.1. Overview of New Clustering Features 79
  5.18.2. Web Session Clustering Changes 80
  5.18.3. Stateful Session EJB Clustering Changes 82
  5.18.4. Clustering Services Changes 83
  5.18.5. Migrate Clustering HA Singleton 83

CHAPTER 6. MISCELLANEOUS CHANGES ......................................................... 84
  6.1. CHANGES TO DELIVERY OF JBOSS EAP NATIVES AND APACHE HTTP SERVER 84
  6.2. CHANGES TO DEPLOYMENTS ON AMAZON EC2 85
  6.3. CHANGES TO JBOSS EAP SCRIPTS 85
  6.4. REMOVAL OF OSGI SUPPORT 85

CHAPTER 7. MIGRATING FROM OLDER RELEASES OF JBOSS EAP ...................... 87
  7.1. MIGRATING FROM JBOSS EAP 5 TO JBOSS EAP 7 87
  7.2. SUMMARY OF CHANGES MADE TO EACH RELEASE 87
  7.3. REVIEW THE CONTENT IN THE MIGRATION GUIDES 88
  7.4. JBOSS EAP 5 COMPONENT UPGRADE REFERENCE 88

APPENDIX A. REFERENCE MATERIAL ................................................................. 95
  A.1. JACORB SUBSYSTEM MIGRATION OPERATION WARNINGS 95
  A.2. MESSAGING SUBSYSTEM MIGRATION OPERATION WARNINGS 96
    Replace the Deprecated broadcast-group or discovery-group Attributes 99
  A.3. WEB SUBSYSTEM MIGRATION OPERATION WARNINGS 100
    Web Subsystem Migration Operation Attribute Warnings 104
    Web SSL Connector Attributes 104
    Web Static Resource Attributes 104
    Web SSO Resource Attributes 105
    Web Access Log Attributes 105
    Web Connector Attributes 105
  A.4. MIGRATE JBOSS WEB SYSTEM PROPERTIES REFERENCE 106
  A.5. COMPATIBILITY AND INTEROPERABILITY BETWEEN RELEASES 115
    EJB remoting over IIOP 115
    EJB remoting Using JNDI 115
Table of Contents

EJB remoting Using @WebService 115
Messaging Standalone Client 115
Messaging MDBs 116
JMS bridges 116
CHAPTER 1. INTRODUCTION

1.1. ABOUT RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 7

Red Hat JBoss Enterprise Application Platform 7 (JBoss EAP) is a middleware platform built on open standards and compliant with the Java Enterprise Edition 7 specification. It integrates WildFly Application Server 10 with messaging, high-availability clustering, and other technologies.

JBoss EAP includes a modular structure that allows service enabling only when required, improving startup speed.

The management console and management command-line interface (CLI) make editing XML configuration files unnecessary and add the ability to script and automate tasks.

JBoss EAP provides two operating modes for JBoss EAP instances: standalone server or managed domain. The standalone server operating mode represents running JBoss EAP as a single server instance. The managed domain operating mode allows for the management of multiple JBoss EAP instances from a single control point.

In addition, JBoss EAP includes APIs and development frameworks for quickly developing secure and scalable Java EE applications.

1.2. ABOUT THE MIGRATION GUIDE

The purpose of this guide is to document the changes that are required to successfully run and deploy Red Hat JBoss Enterprise Application Platform 6 applications on Red Hat JBoss Enterprise Application Platform 7. It provides information about the new features available in this release, the deprecated and unsupported features, and any application and server configuration updates that might be required to prevent changes in application behavior.

It also provides information about tools that can help with the migration, such as Red Hat Application Migration Toolkit, which simplifies migration of Java applications, and the JBoss Server Migration Tool, which updates the server configuration.

Once the application is successfully deployed and running, plans can be made to upgrade individual components to use the new functions and features of JBoss EAP 7.

If you plan to migrate your JBoss EAP 5 applications directly to JBoss EAP 7, see Migrating from Older Releases of JBoss EAP.

1.3. ABOUT MIGRATIONS AND UPGRADES

Major Upgrades

A major upgrade or migration is required when an application is moved from one major release to another, for example, from JBoss EAP 6 to JBoss EAP 7. This is the type of migration addressed in this guide. If an application follows the Java EE specifications, does not access deprecated APIs, and does not contain proprietary code, it might be possible to run the application in JBoss EAP 7 without any application code changes. However, server configuration has changed in JBoss EAP 7 and any server configuration settings require migration.

Minor Updates
JBoss EAP periodically provides point releases, which are minor updates that include bug fixes, security fixes, and new features. The JBoss EAP Patching and Upgrading Guide describes how to upgrade from one point release to another, for example from JBoss EAP 7.0 to JBoss EAP 7.1.

Cumulative Patches

JBoss EAP also periodically provides cumulative patches that contain bug and security fixes. Cumulative patches increment the release by the last digit, for example from 7.0.0 to 7.0.1. Patch installation is covered in detail in the JBoss EAP Patching and Upgrading Guide.

1.4. ABOUT THE USE OF EAP_HOME IN THIS DOCUMENT

In this document, the variable EAP_HOME is used to denote the path to the JBoss EAP installation. Replace this variable with the actual path to your JBoss EAP installation.

- If you installed JBoss EAP using the ZIP install method, the install directory is the jboss-eap-7.0 directory where you extracted the ZIP archive.

- If you installed JBoss EAP using the RPM install method, the install directory is /opt/rh/eap7/root/usr/share/wildfly/.

- If you used the installer to install JBoss EAP, the default path for EAP_HOME is ${user.home}/EAP-7.0.0:
  
  - For Red Hat Enterprise Linux, Solaris, and HP-UX: /home/USER_NAME/EAP-7.0.0/
  - For Microsoft Windows: C:\Users\USER_NAME\EAP-7.0.0\

- If you used the JBoss Developer Studio installer to install and configure the JBoss EAP server, the default path for EAP_HOME is ${user.home}/jbdevstudio/runtimes/jboss-eap:
  
  - For Red Hat Enterprise Linux: /home/USER_NAME/jbdevstudio/runtimes/jboss-eap/
  - For Microsoft Windows: C:\Users\USER_NAME\jbdevstudio\runtimes\jboss-eap or C:\Documents and Settings\USER_NAME\jbdevstudio\runtimes\jboss-eap\

**NOTE**

EAP_HOME is not an environment variable. JBOSS_HOME is the environment variable used in scripts.
CHAPTER 2. PREPARE FOR MIGRATION

2.1. PREPARATION OVERVIEW

In JBoss EAP 7, an effort was made to provide backward compatibility for JBoss EAP 6 applications. However, if your application uses features that were deprecated or functionality that was removed from JBoss EAP 7, you might need to make changes to your application code.

In addition, a number of things have changed in this release that might impact deployment of JBoss EAP 7 applications. It is recommended that you do some research and planning before you attempt to migrate your application.

- Become familiar with the features of Java EE 7.
- Review what’s new in JBoss EAP 7.
- Review the list of deprecated and unsupported features.
- Review the material in the JBoss EAP 7 Getting Started Guide.
- Take a look at the tools that can help with migration tasks.

Once you are comfortable with the feature changes, the development materials, and the tools that can assist your migration efforts, you can begin to evaluate your applications and your server configuration to determine the changes that are needed to run in JBoss EAP 7.

2.2. REVIEW THE JAVA EE 7 FEATURES

Java EE 7 includes many improvements to make it easier to develop and run feature rich applications on private and public clouds. It incorporates new features and the latest standards such as HTML5, WebSocket, JSON, Batch, and Concurrency Utilities. Updates include JPA 2.1, JAX-RS 2.0, Servlet 3.1, Expression Language 3.0, JMS 2.0, JSF 2.2, EJB 3.2, CDI 1.2, and Bean Validation 1.1.

You can find more information about Java EE 7, including tutorials, on Oracle’s web site: Java EE at a Glance

2.3. REVIEW WHAT’S NEW IN JBOSS EAP 7

JBoss EAP 7 includes some notable upgrades and improvements over the previous release.

Java EE 7

JBoss EAP 7 is a certified implementation of Java EE 7, meeting both the Web and the Full profiles. It also includes support for the latest iterations of CDI 1.2 and Web Sockets 1.1.

Undertow

Undertow is the new lightweight, flexible, and performant web server included in JBoss EAP 7, replacing JBoss Web. Written in Java, it is designed for maximum throughput and scalability. It supports the latest web technologies, such as the new HTTP/2 standard.

Apache ActiveMQ Artemis

Apache ActiveMQ Artemis is the new JBoss EAP 7 built-in messaging provider. Based on a code donation from HornetQ, this Apache subproject provides outstanding performance based on a proven non-blocking architecture.

IronJacamar 1.2
The latest IronJacamar provides a stable and feature rich support for JCA and DataSources.

**JBossWS 5**

The fifth generation of JBossWS is a major leap forward, bringing new features and performance improvements to JBoss EAP 7 web services.

**RESTEasy 3**

JBoss EAP 7 includes the latest generation of RESTEasy. It goes beyond the standard Java EE REST APIs (JAX-RS 2.0) by providing a number of useful extensions such as JSON Web Encryption, Jackson, JSON-P, and Jettison.

**OpenJDK ORB**

JBoss EAP 7 replaced the JacORB IIOP implementation with a downstream branch of the OpenJDK ORB, leading to better interoperability with the JVM ORB and the Java EE RI.

**Feature Rich Clustering**

Clustering support was heavily refactored in JBoss EAP 7 and includes several public APIs for access by applications.

**Port Reduction**

By utilizing HTTP upgrade, JBoss EAP 7 has moved nearly all of its protocols to be multiplexed over just two HTTP ports: a management port (9990), and an application port (8080).

**Enhanced Logging**

The management API now supports the ability to list and view the available log files on a server, or even define custom formatters other than the default pattern formatter. Deployment’s logging setup is also greatly enhanced.

For a complete list of new features, see New Features and Enhancements in the JBoss EAP 7 Release Notes.

### 2.4. REVIEW THE LIST OF DEPRECATED AND UNSUPPORTED FEATURES

Before you migrate your application, be aware that some features that were available in previous releases of JBoss EAP might be deprecated or no longer supported.

Support for some technologies was removed due to the high maintenance cost, low community interest, and much better alternative solutions. The following is a short summary of some of the unsupported features.

**EJB Entity Beans**

EJB entity beans are no longer supported. If your application uses EJB entity beans, you should migrate the code to use JPA, which offers a much more performant and flexible API.

**JAX-RPC**

Because JAX-WS offers a much more accurate and complete solution, code written for JAX-RPC should be migrated to use JAX-WS.

**JSR-88**

Java EE Application Deployment API specification (JSR-88), which defined a contract to enable tools from multiple providers to configure and deploy applications on any Java EE platform product, was not widely adopted. You must use another JBoss EAP supported option for application deployment, such as the management console, the management CLI, deployment scanner, or Maven.

**Generic JMS Resource Adapter**
The ability to configure a generic JMS resource adapter to connect to a JMS provider is no longer supported in JBoss EAP 7.

For a comprehensive list of deprecated and unsupported features, see Unsupported and Deprecated Functionality in the JBoss EAP 7 Release Notes.

2.5. REVIEW THE JBOSS EAP GETTING STARTED MATERIAL

Be sure to review the JBoss EAP Getting Started Guide. It contains the following important information:

- How to download and install JBoss EAP 7
- How to download and install Red Hat JBoss Developer Studio
- How to configure Maven for your development environment, manage project dependencies, and configure your projects to use the JBoss EAP Bill of Material (BOM) artifacts
- How to download and run the quickstart example applications that ship with the product

2.6. MIGRATION ANALYSIS AND PLANNING

Each application and server configuration is unique and you must thoroughly understand the components and architecture of the existing application and server platform before you attempt the migration. Your migration plan should include a detailed roadmap for testing and roll out to production that takes into account the following information.

Identify the People Responsible for the Migration

Identify the stakeholders, project managers, developers, administrators, and others who will be responsible for the migration.

Review the Application Server Platform Configuration and Hardware

Examine the existing application server and platform configuration to determine how they are impacted by feature changes in JBoss EAP 7. The review should include the following items.

- Operating systems and versions
- Database used by the applications
- Web servers
- Security architecture
- Number and type of processors
- Amount of memory
- Amount of physical disk storage
- Migration of database or messaging data
- Other components that might be impacted by the migration

Review the Current Production Environment

You should plan to recreate the production environment as closely as possible for testing and staging the migration process.
- Take into account any clustering configurations. See Upgrading a Cluster in the JBoss EAP Patching and Upgrading Guide for more information about how to migrate clusters.

- If you are currently running a large managed domain, consider a gradual migration approach.

- Determine whether you need to migrate any database or messaging data.

Examine and Understand the Existing Application

Thoroughly examine the existing JBoss EAP 6 application. Be totally familiar with its architecture, functions, features and components, including:

- The JVM version
- Integration with other Red Hat application server middleware components
- Integration with proprietary third-party software
- Use of deprecated features that will require replacement
- Application configuration including deployment descriptors, JNDI, persistence, JDBC configuration and pooling, JMS topics and queues, and logging

Identify any code or configuration incompatibilities that will require modification during the migration to JBoss EAP 7.

Create a Detailed Test Plan

- The plan should include regression testing and acceptance criteria requirements.

- It should also include performance testing.

- Set up a staging environment as close to the production environment as possible to test the migration before the rollout to production.

- Be sure to create a backup and backout plan!

Review the Resources Available for the Migration Process

- Assess the skills of the development team and plan for training or additional consulting help.

- Be aware that additional hardware and other resources will be required for staging and testing during the migration process until the effort is completed.

- Determine whether any formal training is needed. If so, add it to the schedule.

Execute the Plan

Gather the necessary resources and implement the migration plan.

IMPORTANT

Before making any modifications to your application, be sure to create a backup copy.

2.7. BACK UP IMPORTANT DATA AND REVIEW SERVER STATE
Before you migrate your application, you need to be aware of the following potential issues.

- The migration might remove temporary folders. Any deployments stored in the `data/content/` directory must be backed up prior to the migration and restored after it completes. Otherwise, the server will fail to start due to the missing content.

- Prior to the migration, handle any open transactions and delete the `data/tx-object-store/` transaction directory.

- The persistent timer data in `data/timer-service-data` must be checked to determine whether it is compatible. Prior to the migration, review the `deployment-*` files in that directory to determine which timers are still in use.

Be sure to also back up the current server configuration and applications before you begin.

### 2.8. MIGRATING AN RPM INSTALLATION

**IMPORTANT**

It is not supported to have more than one RPM-installed instance of JBoss EAP on a single Red Hat Enterprise Linux Server. As a result, we recommend that you migrate your JBoss EAP installation to a new machine when migrating to JBoss EAP 7.

When migrating a JBoss EAP RPM installation from JBoss EAP 6 to JBoss EAP 7, ensure that JBoss EAP 7 is installed on a machine that does not have an existing JBoss EAP RPM installation.

To install JBoss EAP 7 using RPMs, see the JBoss EAP *Installation Guide*.

The migration advice in this guide also applies to migrating RPM installations of JBoss EAP, but you might need to alter some steps (such as how to start JBoss EAP) to suit an RPM installation compared to a ZIP or installer installation.

### 2.9. MIGRATE JBOSS EAP RUNNING AS A SERVICE

If you run JBoss EAP 6 as a service, be sure to review *Configuring JBoss EAP to Run as a Service* in the JBoss EAP *Installation Guide* for updated configuration instructions for JBoss EAP 7.
CHAPTER 3. TOOLS TO ASSIST IN MIGRATION

3.1. USE RHAMT TO ANALYZE APPLICATIONS FOR MIGRATION

Red Hat Application Migration Toolkit (RHAMT) is an extensible and customizable rule-based set of tools that helps simplify migration of Java applications. It analyzes the APIs, technologies, and architectures used by the applications you plan to migrate and provides detailed migration reports for each application. These reports provide the following information.

- Detailed explanations of the migration changes needed.
- Whether the reported change is mandatory or optional.
- Whether the reported change is complex or trivial.
- Links to the code requiring the migration change.
- Hints and links to information about how to make the required changes.
- An estimate of the level of effort for each migration issue found and the total estimated effort to migrate the application.

You can use RHAMT to analyze the code and architecture of your JBoss EAP 6 applications before you migrate them to JBoss EAP 7. The RHAMT rule set for migration from JBoss EAP 6 to JBoss EAP 7 reports on XML descriptors and specific application code and parameters that need to be replaced by an alternative configuration when migrating to JBoss EAP 7.

For more information about how to use Red Hat Application Migration Toolkit to analyze your JBoss EAP 6 applications, see the Red Hat Application Migration Toolkit Getting Started Guide.

3.2. USE THE JBOSS SERVER MIGRATION TOOL TO MIGRATE SERVER CONFIGURATIONS

You can use the JBoss Server Migration Tool to update your configuration to include the new features and settings in JBoss EAP 7 while keeping your existing configuration.

The JBoss Server Migration Tool reads your existing JBoss EAP 6 server configuration file and adds configurations for the new subsystems, updates existing subsystem configurations with new features, and removes obsolete subsystem configurations.

The binary distribution of JBoss Server Migration Tool 1.0.0.Final is available for download from https://github.com/wildfly/wildfly-server-migration/releases. This version supports the migration of standalone servers and managed domains from JBoss EAP 6.4 to JBoss EAP 7.0. For general information about how to run the tool, see the JBoss Server Migration Tool User Guide.

NOTE

The JBoss Server Migration Tool is not supported in JBoss EAP 7.0.
CHAPTER 4. SERVER CONFIGURATION CHANGES

4.1. SERVER CONFIGURATION MIGRATION OPTIONS

To migrate your server configuration from JBoss EAP 6 to JBoss EAP 7, you can either use the JBoss Server Migration Tool or do a manual migration with the help of the management CLI migrate operation.

JBoss Server Migration Tool
You can use the JBoss Server Migration Tool to update your configuration to include the new features and settings in JBoss EAP 7 while keeping your existing configuration. See Use the JBoss Server Migration Tool to Migrate Server Configurations for more information.

Management CLI Migrate Operation
You can use the management CLI migrate operation to update the jacorb, messaging, and web subsystems in the JBoss EAP 6 server configuration file to allow them run on the new release, but be aware that the result is not a complete JBoss EAP 7 configuration. For example:

- The operation does not update the original remote protocol and port settings to the new http-remoting and new port settings now used in JBoss EAP 7.
- The configuration does not include the new JBoss EAP subsystems or features such as clustered singleton deployments, or graceful shutdown.
- The configuration does not include the new Java EE 7 features such as batch processing.
- The migrate operation does not migrate the ejb3 subsystem configuration. For information about possible EJB migration issues, see EJB Server Configuration Changes.

For more information about using the migrate operation to migration the server configuration, see Management CLI Migration Operation.

4.2. MANAGEMENT CLI MIGRATION OPERATION

You can use the management CLI to update your JBoss EAP 6 server configuration files to run on JBoss EAP 7. The management CLI provides a migrate operation to automatically update the jacorb, messaging, and web subsystems from the previous release to the new configuration. You can also execute the describe-migration operation for the jacorb, messaging, and web subsystems to review the proposed migration configuration changes before you perform the migration. There are no replacements for the cmp, jaxr, or threads subsystems and they must be removed from the server configuration.

IMPORTANT
See Server Configuration Migration Options for limitations of the migrate operation. You can also use the JBoss Server Migration Tool to update your configuration to include the new features and settings in JBoss EAP 7 while keeping your existing configuration. See Use the JBoss Server Migration Tool to Migrate Server Configurations for more information.

Table 4.1. Subsystem Migration and Management CLI Operation
### Start the Server and the Management CLI

Follow the steps below to update your JBoss EAP 6 server configuration to run on JBoss EAP 7.

1. Before you begin, review [Back Up Important Data and Review Server State](#). It contains important information about making sure the server is in a good state and the appropriate files are backed up.

2. Start the JBoss EAP 7 server with the JBoss EAP 6 configuration.
   
   a. Back up the JBoss EAP 7 server configuration files.
   
   b. Copy the configuration file from the previous release into the JBoss EAP 7 directory.
      
      ```bash
      $ cp EAP6_HOME/standalone/configuration/standalone-full.xml EAP7_HOME/standalone/configuration
      ```
   
   c. Navigate to the JBoss EAP 7 install directory and start the server with the `--admin-only` argument.
      
      ```bash
      $ bin/standalone.sh -c standalone-full.xml --admin-only
      ```
NOTE

You will see the following org.jboss.as.controller.management-operation ERRORs in the server log when you start the server. These errors are expected and indicate that the legacy subsystem configurations must be removed or migrated to JBoss EAP 7.

- WFLYCTL0402: Subsystems [cmp] provided by legacy extension 'org.jboss.as.cmp' are not supported on servers running this version. Both the subsystem and the extension must be removed or migrated before the server will function.

- WFLYCTL0402: Subsystems [jacob] provided by legacy extension 'org.jboss.as.jacob' are not supported on servers running this version. Both the subsystem and the extension must be removed or migrated before the server will function.

- WFLYCTL0402: Subsystems [jxjr] provided by legacy extension 'org.jboss.as.jxjr' are not supported on servers running this version. Both the subsystem and the extension must be removed or migrated before the server will function.

- WFLYCTL0402: Subsystems [messaging] provided by legacy extension 'org.jboss.as.messaging' are not supported on servers running this version. Both the subsystem and the extension must be removed or migrated before the server will function.

- WFLYCTL0402: Subsystems [threads] provided by legacy extension 'org.jboss.as.threads' are not supported on servers running this version. Both the subsystem and the extension must be removed or migrated before the server will function.

- WFLYCTL0402: Subsystems [web] provided by legacy extension 'org.jboss.as.web' are not supported on servers running this version. Both the subsystem and the extension must be removed or migrated before the server will function.

3. Open a new terminal, navigate to the JBoss EAP 7 install directory, and start the management CLI using the --controller=remote://localhost:9999 arguments.

   $ bin/jboss-cli.sh --connect --controller=remote://localhost:9999

Migrate the JacORB, Messaging, and Web Subsystems

1. To review the configuration changes that will be made to the subsystem before you perform the migration, execute the describe-migration operation.

   The describe-migration operation uses the following syntax.

   /subsystem=SUBSYSTEM_NAME:describe-migration

   The following example describes the configuration changes that will be made to the JBoss EAP 6.4 standalone-full.xml configuration file when it is migrated to JBoss EAP 7. Entries were removed from the output to improve readability and to save space.

   **Example: describe-migration Operation**
2. Execute the `migrate` operation to migrate the subsystem configuration to the subsystem that replaces it in JBoss EAP 7. The operation uses the following syntax.

```
/subsystem=SUBSYSTEM_NAME:migrate
```

**NOTE**

The `messaging` subsystem `describe-migration` and `migrate` operations allow you to pass an argument to configure access by legacy clients. For more information about the command syntax, see [Messaging Subsystem Migration and Forward Compatibility](#).

3. Review the outcome and result of the command. Be sure the operation completed successfully and there are no "migration-warning" entries. This means the migration configuration for the subsystem is complete.

**Example: Successful `migrate` Operation with No Warnings**

```
[standalone@localhost:9999 /] /subsystem=messaging:migrate
{
    "outcome" => "success",
    "result" => {
        "migration-warnings" => [],
        "migration-operations" => [
            {
                "operation" => "add",
                "address" => [("extension" => "org.wildfly.extension.messaging-activemq")],
                "module" => "org.wildfly.extension.messaging-activemq"
            },
            {
                "operation" => "add",
                "address" => [("subsystem" => "messaging-activemq")]
            },
            <!-- *** Entries removed for readability *** -->
            {
                "operation" => "remove",
                "address" => [("subsystem" => "messaging")]
            },
            {
                "operation" => "remove",
                "address" => [("extension" => "org.jboss.as.messaging")]
            }
        ]
    }
}```
If you see "migration-warnings" entries in the log, this indicates the migration of the server configuration completed successfully but it was not able to migrate all of elements and attributes. You must follow the suggestions provided by the "migration-warnings" and run additional management CLI commands to modify those configurations. The following is an example of a migrate operation that returns "migration-warnings".

**Example: migrate Operation with Warnings**

```
[standalone@localhost:9999 /] /subsystem=messaging:migrate
{
   "outcome" => "success",
   "result" => {
      "migration-warnings" => [
         "WFLYMSG0080: Could not migrate attribute group-address from resource [
            ("subsystem" => "messaging-activemq"),
            ("server" => "default"),
            ("broadcast-group" => "groupB")
         ]. Use instead the socket-binding attribute to configure this broadcast-group."
         "WFLYMSG0080: Could not migrate attribute group-port from resource [
            ("subsystem" => "messaging-activemq"),
            ("server" => "default"),
            ("broadcast-group" => "groupB")
         ]. Use instead the socket-binding attribute to configure this broadcast-group."
         "WFLYMSG0080: Could not migrate attribute local-bind-address from resource [
            ("subsystem" => "messaging-activemq"),
            ("server" => "default"),
            ("broadcast-group" => "groupA")
         ]. Use instead the socket-binding attribute to configure this broadcast-group."
         "WFLYMSG0080: Could not migrate attribute local-bind-port from resource [
            ("subsystem" => "messaging-activemq"),
            ("server" => "default"),
            ("broadcast-group" => "groupA")
         ]. Use instead the socket-binding attribute to configure this broadcast-group."
         "WFLYMSG0080: Could not migrate attribute group-address from resource [
            ("subsystem" => "messaging-activemq"),
            ("server" => "default"),
            ("broadcast-group" => "groupA")
         ]. Use instead the socket-binding attribute to configure this broadcast-group."
         "WFLYMSG0080: Could not migrate attribute group-port from resource [
            ("subsystem" => "messaging-activemq"),
            ("server" => "default"),
            ("broadcast-group" => "groupA")
         ]. Use instead the socket-binding attribute to configure this broadcast-group."
      ]
   }
}```
NOTE

The list of migrate and describe-migration warnings for each subsystem is located in the Reference Material at the end of this guide.

- Jacorb Subsystem Migration Operation Warnings
- Messaging Subsystem Migration Operation Warnings
- Web Subsystem Migration Operation Warnings

4. Review the server configuration file to verify the extension, subsystem, and namespace were updated and the existing subsystem configuration was migrated to JBoss EAP 7.

NOTE

You must repeat this process for each of the jacorb, messaging, and web subsystems using the following commands.

/subsystem=jacorb:migrate
/subsystem=messaging:migrate
/subsystem=web:migrate

5. Remove the cmp, jaxr, and threads subsystems and extensions from the server configuration.

While still in the management CLI prompt, remove the obsolete cmp, jaxr, and threads subsystems by executing the following commands.

/subsystem=cmp:remove
/subsystem=jaxr:remove
/subsystem=threads:remove

IMPORTANT

You must migrate the messaging, jacorb, and web subsystems and remove the cmp, jaxr, and threads extensions and subsystems before you can restart the server for normal operation. If you need to restart the server before you complete this process, be sure to include the --admin-only argument on the server start command line. This allows you to continue with the configuration changes.

4.3. LOGGING MESSAGE PREFIX CHANGES

Log messages are prefixed with the project code for the subsystem that reports the message. The prefixes for all log messages have changed in JBoss EAP 7.
For a complete list of the new log message project code prefixes used in JBoss EAP 7, see Project Codes Used in JBoss EAP in the JBoss EAP Development Guide.

4.4. WEB SERVER CONFIGURATION CHANGES

4.4.1. Replace the Web Subsystem with Undertow

Undertow replaces JBoss Web as the web server in JBoss EAP 7. This means the legacy web subsystem configuration must be migrated to the new JBoss EAP 7 undertow subsystem configuration.

- The urn:jboss:domain:web:2.2 subsystem configuration namespace in the server configuration file has been replaced by the urn:jboss:domain:undertow:3.1 namespace.

- The org.jboss.as.web extension module, located in EAP_HOME/modules/system/layers/base/, has been replaced with the org.wildfly.extension.undertow extension module.

You can use the management CLI migrate operation to migrate the web subsystem to undertow in the server configuration file. However, be aware that this operation is not able to migrate all JBoss Web subsystem configurations. If you see "migration-warning" entries, you must run additional management CLI commands to migrate those configurations to Undertow. For more information about the management CLI migrate operation, see Management CLI Migration Operation.

The following is an example of the default web subsystem configuration in JBoss EAP 6.

```xml
<subsystem xmlns="urn:jboss:domain:web:2.2" default-virtual-server="default-host" native="false">
  <connector name="http" protocol="HTTP/1.1" scheme="http" socket-bindings="http"/>
  <virtual-server name="default-host" enable-welcome-root="true">
    <alias name="localhost"/>
    <alias name="example.com"/>
  </virtual-server>
</subsystem>
```

The following is an example of the default undertow subsystem configuration in JBoss EAP 7.

```xml
<subsystem xmlns="urn:jboss:domain:undertow:3.1">
  <buffer-cache name="default"/>
  <server name="default-server">
    <http-listener name="default" socket-binding="http" redirect-socket="https"/>
    <host name="default-host" alias="localhost">
      <location name="/" handler="welcome-content"/>
      <filter-ref name="server-header"/>
      <filter-ref name="x-powered-by-header"/>
    </host>
  </server>
</subsystem>
```
4.4.2. Migrate JBoss Web Rewrite Conditions

The management CLI `migrate` operation is not able to automatically migrate rewrite conditions. They are reported as "migration-warnings" and you must migrate them manually. You can create the equivalent configuration in JBoss EAP 7 by using `Undertow Predicates Attributes and Handlers`.

The following is an example of a `web` subsystem configuration in JBoss EAP 6 that includes rewrite configuration.

```xml
<subsystem xmlns="urn:jboss:domain:web:2.2" default-virtual-server="default" native="false">
  <virtual-server name="default" enable-welcome-root="true">
    <alias name="localhost"/>
    <rewrite name="test" pattern="(\*)/toberewritten/(.*)" substitution="$1/rewritten/$2" flags="NC"/>
    <condition name="get" test="%{REQUEST_METHOD}" pattern="GET"/>
    <condition name="andCond" test="%{REQUEST_URI}" pattern=".*index.html" flags="NC"/>
  </rewrite>
</virtual-server>
</subsystem>
```

Follow the Management CLI Migration Operation instructions to start your server and the management CLI, then migrate the `web` subsystem configuration file using the following command.

```
/subsystem=web:migrate
```

The following "migration-warnings" are reported when you run the `migrate` operation on the above configuration.

```
"outcome" => "success",
"result" => {
  "migration-warnings" => [
    "WFLYWEB0002: Could not migrate resource {
      "pattern" => "(\*)\\",
      "substitution" => "\\-\\",
      "flags" => "F",
      "operation" => "add",
      "address" => ["subsystem" => "web"],
    }
  ]
}
```

Red Hat JBoss Enterprise Application Platform 7.0 Migration Guide
(<"virtual-server" => "default-host"),
(<"rewrite" => "test2")
]
"WFLYWEB0002: Could not migrate resource {
(<"test" => "%{REQUEST_METHOD}"),
(<"pattern" => "GET"),
(<"flags" => undefined,
(<"operation" => "add"),
(<"address" => [ 
( (<"subsystem" => "web"),
(<"virtual-server" => "default-host"),
(<"rewrite" => "test2"),
(<"condition" => "get")
]
"WFLYWEB0002: Could not migrate resource {
(<"test" => "%{REQUEST_URI}"),
(<"pattern" => ".*index.html"),
(<"flags" => "NC"),
(<"operation" => "add"),
(<"address" => [ 
( (<"subsystem" => "web"),
(<"virtual-server" => "default-host"),
(<"rewrite" => "test2"),
(<"condition" => "andCond")
]
}])
"

Review the server configuration file and you see the following configuration for the undertow subsystem.

NOTE

The rewrite configuration is dropped.

<subsystem xmlns="urn:jboss:domain:undertow:3.1">
  <buffer-cache name="default"/>
  <server name="default-server">
    <http-listener name="http" socket-binding="http"/>
    <host name="default-host" alias="localhost, example.com">
      <location name="/" handler="welcome-content"/>
    </host>
  </server>
  <servlet-container name="default">
    <jsp-config/>
  </servlet-container>
  <handlers>
    <file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
  </handlers>
</subsystem>
Use the management CLI to create the filter to replace the rewrite configuration in the `undertow` subsystem. You should see "("outcome" ⇒ "success")" for each command.

```bash
# Create the filters
/subsystem=undertow/configuration=filter/expression-filter="test1":add(expression="path('(.*)/toberewritten/(.*)') -> rewrite('$1/rewritten/$2')")
/subsystem=undertow/configuration=filter/expression-filter="test2":add(expression="method('GET') and path('.*index.html') -> response-code(403)")

# Add the filters to the default server
/subsystem=undertow/server=default-server/host=default-host/filter-ref="test1":add
/subsystem=undertow/server=default-server/host=default-host/filter-ref="test2":add
```

Review the updated server configuration file. The JBoss Web subsystem is now completely migrated and configured in the `undertow` subsystem.

```xml
<subsystem xmlns="urn:jboss:domain:undertow:3.1">
  <buffer-cache name="default"/>
  <server name="default-server">
    <http-listener name="http" socket-binding="http"/>
    <host name="default-host" alias="localhost, example.com">
      <location name="/" handler="welcome-content"/>
      <filter-ref name="test1"/>
      <filter-ref name="test2"/>
    </host>
  </server>
  <servlet-container name="default">
    <jsp-config/>
  </servlet-container>
  <handlers>
    <file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
  </handlers>
  <filters>
    <expression-filter name="test1" expression="path('(.*)/toberewritten/(.*)') -> rewrite('$1/rewritten/$2')"/>
    <expression-filter name="test2" expression="method('GET') and path('.*index.html') -> response-code(403)"/>
  </filters>
</subsystem>
```

For more information about how to configure filters and handlers using the management CLI, see Configuring the Web Server in the JBoss EAP 7 Configuration Guide.

### 4.4.3. Migrate JBoss Web System Properties

In the previous release of JBoss EAP, system properties could be used to modify the default JBoss Web behavior. For information about how to configure the same behavior in Undertow, see JBoss Web System Properties Migration Reference.
4.4.4. Migrate JBoss Web jboss-web.xml Overlay

In JBoss EAP 6, JBoss Web supported the ability to specify additional static files for a web application using the overlay element in the jboss-web.xml file. This feature did not actually overlay existing files, but instead added static files to a deployment outside of the WAR archive.

In the following jboss-web.xml file example, /tmp/mycontents/test.html was returned by the JBoss EAP 6 server when you accessed the URL http://localhost:8080/example/test.html.

```xml
<jboss-web>
  <context-root>/example</context-root>
  <overlay>/tmp/mycontents/</overlay>
</jboss-web>
```

Undertow, which replaces JBoss Web in JBoss EAP 7, does not support the jboss-web.xml file overlay feature.

**IMPORTANT**

This overlay feature is available if you apply Red Hat JBoss Enterprise Application Platform 7.0 Update 05 or a newer cumulative patch to your JBoss EAP installation.

4.4.5. Migrate Global Valves

Previous releases of JBoss EAP supported valves. Valves are custom classes inserted into the request processing pipeline for an application before servlet filters to make changes to the request or perform additional processing.

- Global valves are inserted into the request processing pipeline of all deployed applications and are configured in the server configuration file.
- Authenticator valves authenticate the credentials of the request.
- Custom application valves were created by extending the org.apache.catalina.valves.ValveBase class and configured in the <valve> element of the jboss-web.xml descriptor file. These valves must be migrated manually.

This section describes how to migrate global valves. Migration of custom and authenticator valves are covered in the Migrate Custom Application Valves section of this guide.

Undertow, which replaces JBoss Web in JBoss EAP 7, does not support global valves; however, you should be able to achieve similar functionality by using Undertow handlers. Undertow includes a number of built-in handlers that provide common functionality. It also provides the ability to create custom handlers, which can be used to replace custom valve functionality.

If your application uses valves, you must replace them with the appropriate Undertow handler code to achieve the same functionality when you migrate to JBoss EAP 7.

For more information about how to configure handlers, see Configuring Handlers in the JBoss EAP 7 Configuration Guide.

For more information about how to configure filters, see Configuring Filters in the JBoss EAP 7 Configuration Guide.

Migrate JBoss Web Valves

The following table lists the valves that were provided by JBoss Web in the previous release of JBoss.
EAP and the corresponding Undertow built-in handler. The JBoss Web valves are located in the org.apache.catalina.valves package.

### Table 4.2. Mapping Valves to Handlers

<table>
<thead>
<tr>
<th>Valve</th>
<th>Handler</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessLogValve</td>
<td>io.undertow.server.handlers.accesslog.AccessLogHandler</td>
</tr>
<tr>
<td>CrawlerSessionManagerValve</td>
<td>io.undertow.servlet.handlers.CrawlerSessionManagerHandler</td>
</tr>
<tr>
<td>ExtendedAccessLogValve</td>
<td>io.undertow.server.handlers.accesslog.AccessLogHandler</td>
</tr>
<tr>
<td>JDBCAccessLogValve</td>
<td>See the JDBCAccessLogValve Manual Migration Procedure below for instructions.</td>
</tr>
<tr>
<td>RemoteAddrValve</td>
<td>io.undertow.server.handlers.IPAddressAccessControlHandler</td>
</tr>
<tr>
<td>RemoteHostValve</td>
<td>io.undertow.server.handlers.AccessControlListHandler</td>
</tr>
<tr>
<td>RemoteIpValve</td>
<td>io.undertow.server.handlers.ProxyPeerAddressHandler</td>
</tr>
<tr>
<td>RequestDumperValve</td>
<td>io.undertow.server.handlers.RequestDumpingHandler</td>
</tr>
<tr>
<td>RewriteValve</td>
<td>See Migrate JBoss Web Rewrite Conditions for instructions to migrate these valves manually.</td>
</tr>
<tr>
<td>StuckThreadDetectionValve</td>
<td>io.undertow.server.handlers.StuckThreadDetectionHandler</td>
</tr>
</tbody>
</table>

You can use the management CLI migrate operation to automatically migrate global valves that meet the following criteria:

- They are limited to the valves listed in the previous table that do not require manual processing.
- They must be defined in the web subsystem of the server configuration file.

For more information about the management CLI migrate operation, see Management CLI Migration Operation.

### JDBCAccessLogValve Manual Migration Procedure

The org.apache.catalina.valves.JDBCAccessLogValve valve is an exception to the rule and can not be automatically migrated to io.undertow.server.handlers.JDBCLogHandler. Follow the steps below to migrate the following example valve.

```xml
<valve name="jdbc" module="org.jboss.as.web" class-name="org.apache.catalina.valves.JDBCAccessLogValve">
    <param param-name="driverName" param-value="com.mysql.jdbc.Driver"/>
    <param param-name="connectionName" param-value="root"/>
    <param param-name="connectionPassword" param-value="password"/>
    <param param-name="connectionURL" param-
1. Create a driver module for the database that will store the log entries.

2. Configure the datasource for the database and add the driver to the list of available drivers in the datasources subsystem.

3. Configure an expression-filter in the undertow subsystem with the following expression: `jdbc-access-log(datasource='java:jboss/datasources/accessLogDS')`.

4.5. JGROUPS SERVER CONFIGURATION CHANGES

4.5.1. JGroups Defaults to a Private Network Interface

In the JBoss EAP 6 default configuration, JGroups used the public interface defined in the <interfaces> section of the server configuration file.

Because it is a recommended practice to use a dedicated network interface, by default, JGroups now uses the new private interface that is defined in the <interfaces> section of the server configuration file in JBoss EAP 7.

4.5.2. JGroups Channels Changes
JGroups provides group communication support for HA services in the form of JGroups channels. JBoss EAP 7 introduces `<channel>` elements to the `jgroups` subsystem in the server configuration file. You can add, remove, or change the configuration of JGroups channels using the management CLI.

For more information about how to configure JGroups, see Cluster Communication with JGroups in the JBoss EAP Configuration Guide.

### 4.6. INFINISPAN SERVER CONFIGURATION CHANGES

#### 4.6.1. Infinispan Default Cache Configuration Changes

In JBoss EAP 6, the default clustered caches for web session replication and EJB replication were replicated ASYNC caches. This has changed in JBoss EAP 7. The default clustered caches are now distributed ASYNC caches. The replicated caches are no longer even configured by default. See Configure the Cache Mode in the JBoss EAP Configuration Guide for information about how to add a replicated cache and make it the default.

This only affects you when you use the new JBoss EAP 7 default configuration. If you migrate the configuration from JBoss EAP 6, the configuration of the infinispan subsystem will be preserved.

#### 4.6.2. Infinispan Cache Strategy Changes

The behavior of ASYNC cache strategy has changed in JBoss EAP 7.

In JBoss EAP 6, ASYNC cache reads were lock free. Although they would never block, they were prone to dirty reads of stale data, for example on failover. This is because it would allow subsequent requests for the same user to start before the previous request completed. This permissiveness is not acceptable when using distributed mode, since cluster topology changes can affect session affinity and easily result in stale data.

In JBoss EAP 7, ASYNC cache reads require locks. Since they now block new requests from the same user until the previous replication finishes, dirty reads are prevented.

#### 4.6.3. Configuring Custom Stateful Session Bean Cache for Passivation

Be aware of the following restrictions when configuring a custom stateful session bean (SFSB) cache for passivation in JBoss EAP 7.0.

- The `idle-timeout` attribute, which is configured in the `infinispan passivation-store` of the `ejb3` subsystem, is deprecated in JBoss EAP 7.0. JBoss EAP 6.4 supported eager passivation, passivating according to the `idle-timeout` value. JBoss EAP 7.0 supports lazy passivation, passivating when the `max-size` threshold is reached.

- In JBoss EAP 7.0, the cluster name used by the EJB client is determined by the actual cluster name of the channel, as configured in the `jgroups` subsystem.

- JBoss EAP 7.0 still allows you to set the `max-size` attribute to control the passivation threshold.

- You should not configure eviction or expiration in your EJB cache configuration.
  - You should configure eviction by using the `max-size` attribute of the `passivation-store` in the `ejb3` subsystem.
You should configure expiration by using the @StatefulTimeout annotation in the SFSB Java source code or by specifying a stateful-timeout value in the ejb-jar.xml file.

**4.7. EJB SERVER CONFIGURATION CHANGES**

There is no migrate operation for the ejb3 subsystem, so if you use the management CLI migrate operations to upgrade your other existing JBoss EAP 6.4 configurations, be aware that the ejb3 subsystem configuration is not migrated. Because the configuration of the ejb3 subsystem is slightly different in JBoss EAP 7.0 than in JBoss EAP 6.4, you might see exceptions in the server log when you deploy your EJB applications.

**IMPORTANT**

If you use the JBoss Server Migration Tool to update your server configuration, the ejb3 subsystem should be configured correctly and you should not see any issues when you deploy your EJB applications.

**DuplicateServiceException**
The following DuplicateServiceException is caused by caching changes in JBoss EAP 7.

**DuplicateServiceException in Server Log**

```
... Caused by: org.jboss.msc.service.DuplicateServiceException: Service jboss.infinispan.ejb."mdb-1.0-SNAPSHOT.jar".config is already registered
```

You must reconfigure the cache to resolve this error.

1. Follow the instructions to Start the Server and the Management CLI.

2. Issue the following commands to reconfigure caching in the ejb3 subsystem.

   ```
   /subsystem=ejb3/file-passivation-store=file:remove
   /subsystem=ejb3/cluster-passivation-store=infinispan:remove
   /subsystem=ejb3/passivation-store=infinispan:add(cache-container=ejb, max-size=10000)
   
   /subsystem=ejb3/cache=passivating:remove
   /subsystem=ejb3/cache=clustered:remove
   /subsystem=ejb3/cache=distributable:add(passivation-store=infinispan, aliases=[passivating, clustered])
   ```

**4.8. MESSAGING SERVER CONFIGURATION CHANGES**

In JBoss EAP 7, ActiveMQ Artemis replaces HornetQ as the JMS support provider. This section describes how to migrate the configuration and related messaging data.
4.8.1. Messaging Subsystem Server Configuration Changes

The `org.jboss.as.messaging` module extension, located in `EAP_HOME/modules/system/layers/base/`, has been replaced by the `org.wildfly.extension.messaging-activemq` extension module.

The `urn:jboss:domain:messaging:3.0` subsystem configuration namespace has been replaced by the `urn:jboss:domain:messaging-activemq:1.0` namespace.

Management Model

In most cases, an effort was made to keep the element and attribute names as similar as possible to those used in previous releases. The following table lists some of the changes.

<table>
<thead>
<tr>
<th>HornetQ Name</th>
<th>ActiveMQ Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>hornetq-server</td>
<td>server</td>
</tr>
<tr>
<td>hornetq-serverType</td>
<td>serverType</td>
</tr>
<tr>
<td>connectors</td>
<td>connector</td>
</tr>
<tr>
<td>discovery-group-name</td>
<td>discovery-group</td>
</tr>
</tbody>
</table>

The management operations invoked on the new `messaging-activemq` subsystem have changed from `/subsystem=messaging/hornetq-server=` to `/subsystem=messaging-activemq/server=`.

You can migrate an existing JBoss EAP 6 messaging subsystem configuration to the `messaging-activemq` subsystem on a JBoss EAP 7 server by invoking its `migrate` operation.

```bash
/subsystem=messaging:migrate
```

Before you execute the `migrate` operation, you can invoke the `describe-migration` operation to review the list of management operations that will be performed to migrate from the existing JBoss EAP 6 messaging subsystem configuration to the `messaging-activemq` subsystem on the JBoss EAP 7 server.

```bash
/subsystem=messaging:describe-migration
```

The `migrate` and `describe-migration` operations also display a list of `migration-warnings` for resources or attributes that can not be migrated automatically.

Messaging Subsystem Migration and Forward Compatibility

The `describe-migration` and `migrate` operations for the `messaging` subsystem provide an additional configuration argument. If you want to configure messaging to allow legacy JBoss EAP 6 clients to connect to the JBoss EAP 7 server, you can add the boolean `add-legacy-entries` argument to the `describe-migration` or `migrate` operation as follows.

```bash
/subsystem=messaging:describe-migration(add-legacy-entries=true)
/subsystem=messaging:migrate(add-legacy-entries=true)
```
If the boolean argument `add-legacy-entries` is set to `true`, the `messaging-activemq` subsystem creates the `legacy-connection-factory` resource and adds `legacy-entries` to the `jms-queue` and `jms-topic` resources.

If the boolean argument `add-legacy-entries` is set to `false`, no legacy resources are created in the `messaging-activemq` subsystem and legacy JMS clients will not be able to communicate with the JBoss EAP 7 servers. This is the default value.

For more information about forward and backward compatibility see the Backward and Forward Compatibility in Configuring Messaging for JBoss EAP.

For more information about the management CLI `migrate` and `describe-migration` operations, see Management CLI Migration Operation.

**Change in Behavior of forward-when-no-consumers Attribute**
The behavior of the `forward-when-no-consumers` attribute has changed in JBoss EAP 7.

In JBoss EAP 6, when `forward-when-no-consumers` was set to `false` and there were no consumers in a cluster, messages were redistributed to all nodes in a cluster.

This behavior has changed in JBoss EAP 7. When `forward-when-no-consumers` is set to `false` and there are no consumers in a cluster, messages are not redistributed. Instead, they are kept on the original node to which they were sent.

**Change in Default Cluster Load Balancing Policy**
The default cluster load balancing policy has changed in JBoss EAP 7.

In JBoss EAP 6, the default cluster load balancing policy was similar to `STRICT`, which is like setting the legacy `forward-when-no-consumers` parameter to `true`. In JBoss EAP 7, the default is now `ON_DEMAND`, which is like setting the legacy `forward-when-no-consumers` parameter to `false`. For more information about these settings, see Cluster Connection Attributes in Configuring Messaging for JBoss EAP.

**Messaging Subsystem XML Configuration**
The XML configuration has changed significantly with the new `messaging-activemq` subsystem to provide an XML scheme more consistent with other JBoss EAP subsystems.

It is strongly advised that you do not attempt to modify the JBoss EAP messaging subsystem XML configuration to conform to the new `messaging-activemq` subsystem. Instead, invoke the legacy subsystem `migrate` operation. This operation will write the XML configuration of the new `messaging-activemq` subsystem as a part of its execution.

### 4.8.2. Migrate Messaging Data

Due to the change from HornetQ to ActiveMQ Artemis as the JMS support provider, both the format and the location of the messaging data has changed in JBoss EAP 7.

You can take one of the following two approaches to migrate the data:

- You can export the HornetQ messaging data from the previous release and import it using the management CLI `import-journal` operation. See Migrate Messaging Data Using Export and Import for more information.

- You can migrate the data from the previous release by configuring a JMS bridge. This process is described in Migrate Messaging Data Using a JMS Bridge.
See Mapping Messaging Folder Names for details of the changes to the messaging data folder names between the releases.

### 4.8.2.1. Migrate Messaging Data Using Export and Import

Using this approach, you export the data from the previous release using the HornetQ exporter utility. You then import the XML formatted output file by using the import-journal operation.

**WARNING**
If you are using JBoss EAP 7.0, you must apply Red Hat JBoss Enterprise Application Platform 7.0 Update 05 or a newer cumulative patch to your JBoss EAP installation in order to avoid a known issue when reading large messages. This issue does not affect JBoss EAP 7.1 or later.

For more information, see JBEAP-4407 - Consumer crashes with IndexOutOfBoundsException when reading large messages from imported journal.

---

**Export Messaging Data from the Previous Release**

**IMPORTANT**

The JBoss EAP 6 server must be stopped before you export the messaging data.

The HornetQ exporter utility generates and exports the messaging data from JBoss EAP 6 to an XML formatted file. This command requires that you specify the paths to the required HornetQ JARs that shipped with JBoss EAP 6, pass the paths to messagingbindings/, messagingjournal/, messagingpaging/, and messaginglargemessages/ folders from the previous release as arguments, and specify an output file in which to write the exported XML data.

The following is the syntax required by the HornetQ exporter utility.

```bash
java -jar -mp MODULE_PATH org.hornetq.exporter MESSAGING_BINDINGS_DIRECTORY MESSAGING_JOURNAL_DIRECTORY MESSAGING_PAGING_DIRECTORY MESSAGING_LARGE_MESSAGES_DIRECTORY > OUTPUT_DATA.xml
```

Create a custom module to ensure the correct versions of the HornetQ JARs, including any JARs installed with patches or upgrades, are loaded and made available to the exporter utility. Using your favorite editor, create a new module.xml file in the

EAP6_HOME/modules/org/hornetq/exporter/main/ directory and copy the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="org.hornetq.exporter">
  <main-class name="org.hornetq.jms.persistence.impl.journal.XmlDataExporter"/>
  <properties>
    <property name="jboss.api" value="deprecated"/>
  </properties>
  <dependencies>
```

---

32
The custom module is created in the modules/ directory, not the modules/system/layers/base/ directory.

Follow the steps below to export the data.

1. Stop the JBoss EAP 6 server.
2. Create the custom module as described above.
3. Run the following command to export the data.
   
   ```
   $ java -jar jboss-modules.jar -mp modules/ org.hornetq.exporter standalone/data/messagingbindings/ standalone/data/messagingjournal/
   standalone/data/messagingpaging
   standalone/data/messaginglargemessages/ > OUTPUT_DIRECTORY/OldMessagingData.xml
   ```
4. Make sure there are no errors or warning messages in the log at the completion of the command.
5. Use tooling available for your operating system to validate the XML in the generated output file.

Import the XML Formatted Messaging Data

You then import the XML formatted output file by using the `import-journal` operation as follows.

**IMPORTANT**

If your target server has already performed some messaging tasks, be sure to back up your messaging folders before you begin the `import-journal` operation to prevent data loss in the event of an import failure. See [Backing Up Messaging Folder Data](#) for more information.

1. If you are migrating your JBoss EAP 6.4 server to 7.x, make sure you have completed the migration of the server configuration before you begin by using the `management CLI migrate operation` or by running the [JBoss Server Migration Tool](#).
2. Start the JBoss EAP 7.x server in normal mode with **NO** JMS clients connected.

**IMPORTANT**

It is important that you start the server with no JMS clients connected. This is because the `import-journal` operation behaves like a JMS producer. Messages are immediately available when the operation is in progress. If this operation fails in the middle of the import and JMS clients are connected, there is no way to recover because JMS clients might have already consumed some of the messages.
IMPORTANT

Do NOT run this command more than one time as doing so will result in duplicate messages!

Recovering from an Import Messaging Data Failure
If the import-journal operation fails, you can attempt to recover by using the following steps.

1. Shut down the JBoss EAP 7.x server.

2. Delete all of the messaging journal folders. See Backing Up Messaging Folder Data for the management CLI commands to determine the correct directory location for the messaging journal folders.

3. If you backed up the target server messaging data prior to the import, copy the messaging folders from the backup location to the messaging journal directory determined in the prior step.

4. Repeat the steps to import the XML formatted messaging data.

4.8.2.2. Migrate Messaging Data Using a JMS Bridge

Using this approach, you configure and deploy a JMS bridge to the JBoss EAP 7 server that moves messages from the JBoss EAP 6 HornetQ queue to the JBoss EAP 7 ActiveMQ Artemis queue.

A JMS bridge consumes messages from a source JMS queue or topic and sends them to a target JMS queue or topic, which is typically on a different server. It can be used to bridge messages between any JMS servers, as long as they are JMS 1.1 compliant. The source and destination JMS resources are looked up using JNDI and the client classes for the JNDI lookup must be bundled in a module. The module name is then declared in the JMS bridge configuration.

This section describes how to configure the servers and deploy a JMS bridge to move the messaging data from JBoss EAP 6 to JBoss EAP 7.

Configure the JBoss EAP 6 Server

1. Stop the JBoss EAP 6 server.

2. Back up the HornetQ journal and configuration files.

   - By default, the HornetQ journal is located in the EAP6_HOME/standalone/data/ directory.

   - See Mapping Messaging Folder Names for default messaging folder locations for each release.

3. Make sure that the InQueue JMS queue containing the JMS messages is defined on the JBoss EAP 6 server.

4. Make sure that messaging subsystem configuration contains an entry for the RemoteConnectionFactory similar to the following.
CHAPTER 4. SERVER CONFIGURATION CHANGES

<connection-factory name="RemoteConnectionFactory">
  <entries>
    <entry name="java:jboss/exported/jms/RemoteConnectionFactory"/>
  </entries>
  ...
</connection-factory>

If it does not contain the entry, create one using the following management CLI command:

/subsystem=messaging/hornetq-server=default/connection-factory=RemoteConnectionFactory:add(factory-type=XA_GENERIC, connector=[netty], entries=[java:jboss/exported/jms/RemoteConnectionFactory], ha=true, block-on-acknowledge=true, retry-interval=1000, retry-interval-multiplier=1.0, reconnect-attempts=-1)

Configure the JBoss EAP 7 Server

1. The JMS bridge configuration needs the org.hornetq module to connect to the HornetQ server in the previous release. This module and its direct dependencies are not present in JBoss EAP 7, so you must copy the following modules from the previous release.

   - Copy the org.hornetq module into the JBoss EAP 7 EAP_HOME/modules/org/ directory.
     - If you did not apply patches to this module, copy this folder from the JBoss EAP 6.4 server: EAP6_HOME/modules/system/layers/base/org/hornetq/
     - If you did apply patches to this module, copy this folder from the JBoss EAP 6.4 server: EAP6_HOME/modules/system/layers/base/.overlays/layer-base-jboss-eap-6.4.x.CP/org/hornetq/

   - Remove the <resource-root> for the HornetQ lib path from the JBoss EAP 7 EAP_HOME/modules/org/hornetq/main/module.xml file.
     - If you did not apply patches to the JBoss EAP 6.4 org.hornetq module, remove the following line from the file:

```
<resource-root path="lib"/>
```

     - If you did apply patches to the JBoss EAP 6.4 org.hornetq module, remove the following lines from the file:

```
<resource-root path="lib"/>
<resource-root path="../../../org/hornetq/main/lib"/>
```
Failure to remove the HornetQ lib path resource-root will cause the bridge to fail with the following error in the log file.

```
    ("subsystem" => "messaging-activemq"),
    ("jms-bridge" => "myBridge")
]) - failure description: "WFLYMSGAMQ0086: Unable to load module org.hornetq"
```

- Copy the `org.jboss.netty` module into the JBoss EAP 7 `EAP_HOME/modules/org/jboss/` directory.
  - If you did not apply patches to this module, copy this folder from the JBoss EAP 6.4 server: `EAP6_HOME/modules/system/layers/base/org/jboss/netty/`
  - If you did apply patches to this module, copy this folder from the JBoss EAP 6.4 server: `EAP6_HOME/modules/system/layers/base/.overlays/layer-base-jboss-eap-6.4.x.CP/org/jboss/netty`

2. Create the JMS queue to contain the messages received from JBoss EAP 6 server. The following is an example of a management CLI command that creates the `MigratedMessagesQueue` JMS queue to receive the message.

```
jms-queue add --queue-address=MigratedMessagesQueue --entries= [jms/queue/MigratedMessagesQueue java:jboss/exported/jms/queue/MigratedMessagesQueue]
```

This creates the following `jms-queue` configuration for the default server in the `messaging-activemq` subsystem of the JBoss EAP 7 server.

```
<jms-queue name="MigratedMessagesQueue"
    entries="jms/queue/MigratedMessagesQueue java:jboss/exported/jms/queue/MigratedMessagesQueue"/>
```

3. Make sure that `messaging-activemq` subsystem default server contains a configuration for the `InVmConnectionFactory` connection-factory similar to the following:

```
<connection-factory name="InVmConnectionFactory" factory-type="XA_GENERIC" entries="java:/ConnectionFactory connectors="in-vm"/>
```

If it does not contain the entry, create one using the following management CLI command:
4. Create and deploy a JMS bridge that reads messages from the InQueue JMS queue configured on the JBoss EAP 6 server and transfers them to the MigratedMessagesQueue configured on the JBoss EAP 7 server.

This creates the following jms-bridge configuration in the messaging-activemq subsystem of the JBoss EAP 7 server.

5. If security is configured for JBoss EAP 6, you must also configure the JMS bridge configuration <source> element to include a source-context that specifies the correct user name and password to use for the JNDI lookup when creating the connection.

Migrate the Data

1. Verify that the information you provided for the following configurations is correct.
   - Any queue and topic names
   - The java.naming.provider.url for JNDI lookup

2. Make sure that you have deployed the target JMS destination to the JBoss EAP 7 server.

3. Start both the JBoss EAP 6 and JBoss EAP 7 servers.
4.8.2.3. Mapping Messaging Folder Names

The following table shows the messaging directory names used in the previous release and the corresponding names used in the current release of JBoss EAP. The directories are relative to the `jboss.server.data.dir` directory, which defaults to `EAP_HOME/standalone/data/` if it is not specified.

<table>
<thead>
<tr>
<th>JBoss EAP 6 Directory Name</th>
<th>JBoss EAP 7 Directory Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>messagingbindings/</td>
<td>activemq/bindings/</td>
</tr>
<tr>
<td>messagingjournal/</td>
<td>activemq/journal/</td>
</tr>
<tr>
<td>messaginglargemessages/</td>
<td>activemq/largemessages/</td>
</tr>
<tr>
<td>messagingpaging/</td>
<td>activemq/paging/</td>
</tr>
</tbody>
</table>

NOTE

The `messaginglargemessages/` and `messagingpaging/` directories might not be present if there are no large messages or if paging is disabled.

4.8.2.4. Backing Up Messaging Folder Data

If your target server has already processed messages, it is a good idea to back up the target message folders to a backup location before you begin. The default location of the messaging folders is `EAP_HOME/standalone/data/activemq/`; however it is configurable. If you are not sure of the location of your messaging data, you can use the following management CLI commands to find the location of the messaging folders.

```bash
/subsystem=messaging-activemq/server=default/path=journal-directory:resolve-path
/subsystem=messaging-activemq/server=default/path=paging-directory:resolve-path
/subsystem=messaging-activemq/server=default/path=bindings-directory:resolve-path
/subsystem=messaging-activemq/server=default/path=large-messages-directory:resolve-path
```

Once you know the location of the folders, copy each folder to a safe backup location.

4.8.3. Migrate JMS Destinations

In previous releases of JBoss EAP, JMS destination queues were configured in the `<jms-destinations>` element under the `<hornetq-server>` element in the messaging subsystem.

```xml
<hornetq-server>
  ...
  <jms-destinations>
    <jms-queue name="testQueue">
      <entry name="queue/test"/>
      <entry name="java:jboss/exported/jms/queue/test"/>
  </jms-destinations>
</hornetq-server>
```
In JBoss EAP 7, the JMS destination queue is configured in the default <server> element of the messaging-activemq subsystem.

```xml
<server name="default">...
  <jms-queue name="testQueue" entries="queue/test java:jboss/exported/jms/queue/test"/>
...</server>
```

4.8.4. Migrate Messaging Interceptors

Messaging interceptors have changed significantly in JBoss EAP 7 with the replacement of HornetQ with ActiveMQ Artemis as the JMS messaging provider.

The HornetQ messaging subsystem included in the previous release of JBoss EAP required that you install the HornetQ interceptors by adding them to a JAR and then modifying the HornetQ module.xml file.

The messaging-activemq subsystem included in JBoss EAP 7 does not require modification of a module.xml file. User interceptor classes, which now implement the Apache ActiveMQ Artemis Interceptor interface, can now be loaded from any server module. You specify the module from which the interceptor should be loaded in the messaging-activemq subsystem of the server configuration file.

**Example Interceptor Configuration**

```xml
<subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">
  <server name="default">
    ...
    <incoming-interceptors>
      <class name="com.mycompany.incoming.myInterceptor" module="com.mycompany"/>
      <class name="com.othercompany.incoming.myOtherInterceptor" module="com.othercompany"/>
    </incoming-interceptors>
    <outgoing-interceptors>
      <class name="com.mycompany.outgoing.myInterceptor" module="com.mycompany"/>
      <class name="com.othercompany.outgoing.myOtherInterceptor" module="com.othercompany"/>
    </outgoing-interceptors>
  </server>
</subsystem>
```

4.8.5. Replace Netty Servlet Configuration

In JBoss EAP 6, you could configure a servlet engine to work with the Netty Servlet transport. Because ActiveMQ Artemis replaces HornetQ as the built-in messaging provider in JBoss EAP 7, this
configuration is no longer available. You must replace the servlet configuration to use the new built-in messaging http connectors and http acceptors instead.

4.8.6. Changes in JMS Serialization Behavior Between Releases

The `serialVersionUID` of `javax.jms.JMSException` changed between JMS 1.1 and JMS 2.0.0. This means that if an instance of a JMSException, or any of its subclasses, is serialized using JMS 1.1, it cannot be deserialized using JMS 2.0.0. The reverse is also true. If an instance of JMSException is serialized using JMS 2.0.0, it cannot be deserialized using JMS 1.1. In both of these cases, it throws an exception similar to the following:

```
javax.jms.JMSException: javax.jms.JMSException; local class incompatible:
stream classdesc serialVersionUID = 8951994251593378324, local class
serialVersionUID = 2368476267211489441
```

This issue is fixed in the JMS 2.0.1 maintenance release.

The following table details the JMS implementation for each JBoss EAP release.

Table 4.4. JMS Implementation for Each JBoss EAP Release

<table>
<thead>
<tr>
<th>JBoss EAP Version</th>
<th>JMS Implementation</th>
<th>JMS version</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4</td>
<td>HornetQ</td>
<td>JMS 1.1</td>
</tr>
<tr>
<td>7.0</td>
<td>Apache ActiveMQ Artemis</td>
<td>JMS 2.0.0</td>
</tr>
</tbody>
</table>

Be aware that the `serialVersionUID` incompatibility can result in a migration issue in the following situation.

- If you send a message that contains a JMSException using a JBoss EAP 6.4 client, migrate your messaging data to JBoss EAP 7.0, and then attempt to deserialize that message using a JBoss EAP 7.0 client, the deserialization will fail and it will throw an exception. This is because the `serialVersionUID` in JMS 1.1 is not compatible with the one in JMS 2.0.0.

**IMPORTANT**

Red Hat recommends that you consume all JMS 1.1 messages that contain JMSExceptions before migrating messaging data from JBoss EAP 6.4 to JBoss EAP 7.0.

4.9. JMX MANAGEMENT CHANGES

The HornetQ component in JBoss EAP 6 provided its own JMX management; however, it was not recommended and is now deprecated and no longer supported. If you relied on this feature in JBoss EAP 6, you must migrate your management tooling to use either the JBoss EAP management CLI or the JMX management provided with JBoss EAP 7.

You must also upgrade your client libraries to use the `jboss-client.jar` that ships with JBoss EAP 7.

The following is an example of HornetQ JMX management code that was used in JBoss EAP 6.
JMXConnector connector = null;
try {
    HashMap environment = new HashMap();
    String[] credentials = new String[]{"admin", "Password123!
"};
    environment.put(JMXConnector.CREDENTIALS, credentials);

    // The JMX object name pointed to the HornetQ JMX management
    ObjectName objectName = new
    ObjectName("org.hornetq:type=Server, module=JMS");

    // The invoked method name was "listConnectionIDs"
    String[] connections = (String[]) mbeanServer.invoke(objectName,
    "listConnectionIDs", new Object[]{}, new String[]{});
    for (String connection : connections) {
        System.out.println(connection);
    }
} finally {
    if (connector != null) {
        connector.close();
    }
}

The following is an example of the equivalent code needed for ActiveMQ Artemis in JBoss EAP 7.

JMXConnector connector = null;
try {
    HashMap environment = new HashMap();
    String[] credentials = new String[]{"admin", "Password123!
"};
    environment.put(JMXConnector.CREDENTIALS, credentials);

    // ActiveMQ Artemis uses the protocol "remote+http and port "9990"
    JMXServiceURL beanServerUrl = new
    JMXServiceURL("service:jmx:remote+http://127.0.0.1:9990");

    connector = JMXConnectorFactory.connect(beanServerUrl, environment);
    MBeanServerConnection mbeanServer =
    connector.getMBeanServerConnection();

    // The JMX object name points to the new JMX management in the
    'messaging-activemq' subsystem
    ObjectName objectName = new ObjectName("jboss.as:subsystem=messaging-
    activemq, server=default");

    // The invoked method name is now "listConnectionIds"
    String[] connections = (String[]) mbeanServer.invoke(objectName,
    "listConnectionIds", new Object[]{}, new String[]{});
    for (String connection : connections) {
        System.out.println(connection);
    }
}
Notice that the method names and parameters have changed in the new implementation. You can find the new method names in the JConsole by following these steps.

1. Connect to the JConsole using the following command.
   
   ```bash
   EAP_HOME/bin/jconsole.sh
   ```

2. Connect to JBoss EAP local process. Note that it should start with "jboss-modules.jar".

3. In the MBeans tab, choose `jboss.as → messaging-activemq → default → Operations` to display the list of method names and attributes.

### 4.10. ORB SERVER CONFIGURATION CHANGES

The JacORB implementation has been replaced with a downstream branch of the OpenJDK ORB in JBoss EAP 7.

The `org.jboss.as.jacorb` extension module, located in `EAP_HOME/modules/system/layers/base/`, has been replaced by the `org.wildfly.iiop-openjdk` extension module.

The `urn:jboss:domain:jacorb:1.4` subsystem configuration namespace in the server configuration file has been replaced by the `urn:jboss:domain:iiop-openjdk:1.0` namespace.

The following is an example of the default `jacorb` system configuration in JBoss EAP 6.

```xml
<subsystem xmlns="urn:jboss:domain:jacorb:1.4">
   <orb socket-binding="jacorb" ssl-socket-binding="jacorb-ssl">
      <initializers security="identity" transactions="spec"/>
   </orb>
</subsystem>
```

The following is an example of the default `iiop-openjdk` subsystem configuration in JBoss EAP 7.

```xml
<subsystem xmlns="urn:jboss:domain:iiop-openjdk:1.0">
   <orb socket-binding="jacorb" ssl-socket-binding="jacorb-ssl" />
   <initializers security="identity" transactions="spec" />
</subsystem>
```

The new `iiop-openjdk` subsystem configuration accepts only a subset of the legacy elements and attributes. The following is an example of a `jacorb` subsystem configuration in the previous release of JBoss EAP that contains all valid elements and attributes:

```xml
<subsystem xmlns="urn:jboss:domain:jacorb:1.4">
   <orb name="JBoss" print-version="off" use-imr="off" use-bom="off"
    cache-typecodes="off" cache-poa-names="off" giop-minor-version="2" socket-
```
The following element attributes are no longer supported and must be removed.

Table 4.5. Attributes to Remove

<table>
<thead>
<tr>
<th>Element</th>
<th>Unsupported Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;orb&gt;</td>
<td>• client-timeout</td>
</tr>
<tr>
<td></td>
<td>• max-managed-buf-size</td>
</tr>
<tr>
<td></td>
<td>• max-server-connections</td>
</tr>
<tr>
<td></td>
<td>• outbuf-cache-timeout</td>
</tr>
<tr>
<td></td>
<td>• outbuf-size</td>
</tr>
<tr>
<td></td>
<td>• connection retries</td>
</tr>
<tr>
<td></td>
<td>• retry-interval</td>
</tr>
<tr>
<td></td>
<td>• name</td>
</tr>
<tr>
<td></td>
<td>• server-timeout</td>
</tr>
</tbody>
</table>

```xml
<orb binding="jacobr" ssl-socket-binding="jacobr-ssl">
    <connection retries="5" retry-interval="500" client-timeout="0" server-timeout="0">
        <initializers security="off" transactions="spec"/>
        <poa monitoring="off" queue-wait="on" queue-min="10" queue-max="100">
            <request-processors pool-size="10" max-threads="32" num-threads="100" />
        </poa>
    </connection>
</orb>
```
The following on/off attributes are no longer supported and will not be migrated when you run the management CLI migrate operation. If they are set to on, you will get a migration warning. Other on/off attributes that are not mentioned in this table, for example `<security support-ssl="on|off">`, are still supported and will be migrated successfully. The only difference is that their values will be changed from on/off to true/false.

### Table 4.6. Attributes to Turn Off or Remove

<table>
<thead>
<tr>
<th>Element</th>
<th>Unsupported Attributes</th>
</tr>
</thead>
</table>
| `<poa>`   | • queue-min  
            | • queue-max  
            | • pool-size   
            | • max-threads |
| `<orch>`   | • cache-poa-names  
            | • cache-typecodes  
            | • print-version   
            | • use-bom       
            | • use-imr       |
| `<interop>` | (all except sun)  
            | • comet   
            | • iona    
            | • chunk-custom-rmi-valuetypes  
            | • indirection-encoding-disable  
            | • lax-boolean-encoding  
            | • strict-check-on-tc-creation |
| `<poa/>`  | • monitoring  
            | • queue-wait |

### 4.11. MIGRATE THE THREADS SUBSYSTEM CONFIGURATION
The JBoss EAP 6 server configuration included a threads subsystem that was used to manage thread pools across the various subsystems in the server.

The threads subsystem is no longer available in JBoss EAP 7. Instead, each subsystem is responsible for managing its own thread pools.

For information about how to configure thread pools for the infinispan subsystem, see Configure Infinispan Thread Pools in the JBoss EAP Configuration Guide.

For information about how to configure thread pools for the jgroups subsystem, see Configure JGroups Thread Pools in the JBoss EAP Configuration Guide.

In JBoss EAP 6, you configured thread pools for connectors and listeners for the web subsystem by referencing an executor that was defined in the threads subsystem. In JBoss EAP 7, you now configure thread pools for the undertow subsystem by referencing a worker that is defined in the io subsystem. For more information, see Configuring the IO Subsystem in the JBoss EAP Configuration Guide.

For information about about changes to thread pool configuration in the remoting subsystem, see Migrate the Remoting Subsystem Configuration in this guide, and Configuring the Endpoint in the JBoss EAP Configuration Guide.

4.12. MIGRATE THE REMOTING SUBSYSTEM CONFIGURATION

In JBoss EAP 6, you configured the thread pool for the remoting subsystem by setting various worker-* attributes. The worker thread pool is no longer configured in the remoting subsystem in JBoss EAP 7 and if you attempt to modify the existing configuration, you will see the following message.

```
WFLYRMT0022: Worker configuration is no longer used, please use endpoint worker configuration
```

In JBoss EAP 7, the worker thread pool is replaced by an endpoint configuration that references a worker defined in the io subsystem.

For information about how to configure the endpoint, see Configuring the Endpoint in the JBoss EAP Configuration Guide.

4.13. WEBSOCKET SERVER CONFIGURATION CHANGES

To use WebSockets in JBoss EAP 6, you had to enable the non blocking Java NIO2 connector protocol for the http connector in the web subsystem of the JBoss EAP server configuration file using a command similar to the following.

```
```

To use WebSockets in an application, you also had to create a <enable-websockets> element in the application WEB-INF/jboss-web.xml file and set it to true.

In JBoss EAP 7, you no longer need to configure the server for default WebSocket support or configure the application to use it. WebSockets are a requirement of the Java EE 7 specification and the required protocols are configured by default. More complex WebSocket configuration is done in the servlet-
container of the undertow subsystem of the JBoss EAP server configuration file. You can view the available settings using the following command.

```plaintext
/subsystem=undertow/servlet-container=default/setting=websockets:read-resource(recursive=true)
{
   "outcome" => "success",
   "result" => {
      "buffer-pool" => "default",
      "dispatch-to-worker" => true,
      "worker" => "default"
   }
}
```

WebSocket code examples can also be found in the quickstarts that ship with JBoss EAP.

### 4.14. SINGLE SIGN-ON SERVER CHANGES

The infinispan subsystem still provides distributed caching support for HA services in the form of Infinispan caches in JBoss EAP 7; however the caching and distribution of authentication information is handled differently than in previous releases.

In JBoss EAP 6, if single sign-on (SSO) was not provided an Infinispan cache, the cache was not distributed.

In JBoss EAP 7, SSO is distributed automatically when you select the HA profile. When running the HA profile, each host has its own Infinispan cache, which is based on the default cache of the web cache container. This cache stores the relevant session and SSO cookie information for the host. JBoss EAP handles propagation of individual cache information to all hosts. There is no way to specifically assign an Infinispan cache to SSO in JBoss EAP 7.

In JBoss EAP 7, SSO is configured in the undertow subsystem of the server configuration file.

There are no application code changes required for SSO when migrating to JBoss EAP 7.

### 4.15. DATASOURCE CONFIGURATION CHANGES

#### 4.15.1. JDBC Datasource Driver Name

When you configured a datasource in the previous release of JBoss EAP, the value specified for the driver name depended on the number of classes listed in the META-INF/services/java.sql.Driver file contained in the JDBC driver JAR.

**Driver Containing a Single Class**

If the META-INF/services/java.sql.Driver file specified only one class, the driver name value was simply the name of the JDBC driver JAR. This has not changed in JBoss EAP 7.

**Driver Containing Multiple Classes**

In JBoss EAP 6, if there was more than one class listed in META-INF/services/java.sql.Driver file, you specified which class was the driver class by appending its name to the JAR name, along with the major and minor version, in the following format.

```
JAR_NAME + DRIVER_CLASS_NAME + "\" + MAJOR_VERSION + "\" + MINOR_VERSION
```
In JBoss EAP 7, this has changed. You now specify the driver name using the following format.

\[
\text{JAR_NAME} + \_ + \text{DRIVER_CLASS_NAME} + \_ + \text{MAJOR_VERSION} + \_ + \\
\text{MINOR_VERSION}
\]

**NOTE**
An underscore has been added between the JAR_NAME and the DRIVER_CLASS_NAME.

The MySQL 5.1.31 JDBC driver is an example of a driver that contains two classes. The driver class name is `com.mysql.jdbc.Driver`. The following examples demonstrate the differences between how you specify the driver name in the previous and current release of JBoss EAP.

**Example JBoss EAP 6 Driver Name**

mysql-connector-java-5.1.31-bin.jar.com.mysql.jdbc.Driver_5_1

**Example JBoss EAP 7 Driver Name**

mysql-connector-java-5.1.31-bin.jar_com.mysql.jdbc.Driver_5_1

### 4.16. SECURITY SERVER CONFIGURATION CHANGES

For information about Java Security Manager server configuration changes, see Considerations Moving from Previous Versions in How To Configure Server Security for JBoss EAP.

### 4.17. TRANSACTIONS SUBSYSTEM CHANGES

Some Transaction Manager configuration attributes that were available in the transactions subsystem in JBoss EAP 6 have changed in JBoss EAP 7.

**Removed Transactions Subsystem Attributes**

The following table lists the JBoss EAP 6 attributes that were removed from the transactions subsystem in JBoss EAP 7 and the equivalent replacement attributes.

<table>
<thead>
<tr>
<th>Attribute in JBoss EAP 6</th>
<th>Replacement in JBoss EAP 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>object-store-path</td>
</tr>
<tr>
<td>relative-to</td>
<td>object-store-relative-to</td>
</tr>
</tbody>
</table>

**Deprecated Transactions Subsystem Attributes**

The following attributes that were available in the transactions subsystem in JBoss EAP 6 are deprecated in JBoss EAP 7. The deprecated attributes might be removed in a future release of the product. The following table lists the equivalent replacement attributes.
<table>
<thead>
<tr>
<th>Attribute in JBoss EAP 6</th>
<th>Replacement in JBoss EAP 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>use-hornetq-store</td>
<td>use-journal-store</td>
</tr>
<tr>
<td>hornetq-store-enable-async-io-to</td>
<td>journal-store-enable-async-io</td>
</tr>
<tr>
<td>enable-statistics</td>
<td>statistics-enabled</td>
</tr>
</tbody>
</table>

4.18. CHANGES TO MOD_CLUSTER CONFIGURATION

The configuration for static proxy lists in mod_cluster has changed in JBoss EAP 7.

In JBoss EAP 6, you configured the proxy-list attribute, which was a comma-separated list of httpd proxy addresses specified in the format of hostname:port.

The proxy-list attribute is deprecated in JBoss EAP 7. It has been replaced by the proxies attribute, which is a list of outbound socket binding names.

This change impacts how you define a static proxy list, for example, when disabling advertising for mod_cluster. For information about how to disable advertising for mod_cluster, see Disable Advertising for mod_cluster in the JBoss EAP Configuration Guide.

For more information about mod_cluster attributes, see ModCluster Subsystem Attributes in the JBoss EAP Configuration Guide.
CHAPTER 5. APPLICATION MIGRATION CHANGES

5.1. WEB SERVICES APPLICATION CHANGES

JBossWS 5 brings new features and performance improvements to JBoss EAP 7 web services, mainly through upgrades of the Apache CXF, Apache WSS4J, and Apache Santuario components.

5.1.1. JAX-RPC Support Changes

The Java API for XML-based RPC (JAX-RPC) was deprecated in Java EE 6 and is optional in Java EE 7. It is no longer available or supported in JBoss EAP 7. Applications that use JAX-RPC must be migrated to use JAX-WS, which is the current Java EE standard web services framework.

Use of JAX-RPC web services can be identified in any of the following ways:

- The presence of a JAX-RPC mapping file, which is an XML file with the root element `<java-wsdl-mapping>`.
- The presence of a `webservices.xml` XML descriptor file that contains a `<webservice-description>` element, which includes a `<jaxrpc-mapping-file>` child element. The following is an example of `webservices.xml` descriptor file that defines a JAX-RPC web service.

```
<webservices xmlns="http://java.sun.com/xml/ns/j2ee"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="1.1">
  <webservice-description>
    <webservice-description-name>HelloService</webservice-description-name>
    <wsdl-file>WEB-INF/wsdl/HelloService.wsdl</wsdl-file>
    <jaxrpc-mapping-file>WEB-INF/mapping.xml</jaxrpc-mapping-file>
    <port-component>
      <port-component-name>Hello</port-component-name>
      <wsdl-port>HelloPort</wsdl-port>
      <service-endpoint-interface>org.jboss.chap12.hello.Hello</service-endpoint-interface>
      <service-impl-bean>
        <servlet-link>HelloWorldServlet</servlet-link>
      </service-impl-bean>
    </port-component>
  </webservice-description>
</webservices>
```

- The presence of an `ejb-jar.xml` file, which contains a `<service-ref>` that references a JAX-RPC mapping file.

5.1.2. Apache CXF Spring Web Services Changes

In previous releases of JBoss EAP, you could customize the JBossWS and Apache CXF integration by including a `jbossws-cxf.xml` configuration file with the endpoint deployment archive. One use case for this was to configure interceptor chains for web service client and server endpoints on the Apache CXF bus. This integration required Spring to be deployed in the JBoss EAP server.
Spring integration is no longer supported in JBoss EAP 7. Any application that contains a \texttt{jbossws-cxf.xml} descriptor configuration file must be modified to replace the custom configuration defined in that file. While it is still possible to directly access the Apache CXF API, be aware that the application will not be portable.

The suggested approach is to replace Spring custom configurations with the new JBossWS descriptor configuration options where possible. The JBossWS descriptor-based approach provides similar functionality without requiring modification of the client endpoint code. In some cases, you can replace Spring with Context Dependency Injection (CDI).

\textbf{Apache CXF Interceptors}

The JBossWS descriptor provides new configuration options that allow you to declare the interceptors without modifying the client endpoint code. Instead you declare interceptors within predefined client and endpoint configurations by specifying a list of interceptor class names for the \texttt{cxf.interceptors.in} and \texttt{cxf.interceptors.out} properties.

The following is an example of a \texttt{jaxws-endpoint-config.xml} file that declares interceptors using these properties.

```xml
  <endpoint-config>
    <config-name>org.jboss.test.ws.jaxws.cxf.interceptors.EndpointImpl</config-name>
    <property>
      <property-name>cxf.interceptors.in</property-name>
      <property-value>org.jboss.test.ws.jaxws.cxf.interceptors.EndpointInterceptor,org.jboss.test.ws.jaxws.cxf.interceptors.FooInterceptor</property-value>
    </property>
    <property>
      <property-name>cxf.interceptors.out</property-name>
      <property-value>org.jboss.test.ws.jaxws.cxf.interceptors.EndpointCounterInterceptor</property-value>
    </property>
  </endpoint-config>
</jaxws-config>
```

\textbf{Apache CXF Features}

The JBossWS descriptor allows you to declare features within predefined client and endpoint configurations by specifying a list of feature class names for the \texttt{cxf.features} property.

The following is an example of a \texttt{jaxws-endpoint-config.xml} file that declares a feature using this property.

```xml
  <endpoint-config>
    <config-name>org.jboss.test.ws.jaxws.cxf.interceptors.EndpointImpl</config-name>
    <property>
      <property-name>cxf.interceptors.in</property-name>
      <property-value>org.jboss.test.ws.jaxws.cxf.interceptors.EndpointInterceptor,org.jboss.test.ws.jaxws.cxf.interceptors.FooInterceptor</property-value>
    </property>
    <property>
      <property-name>cxf.interceptors.out</property-name>
      <property-value>org.jboss.test.ws.jaxws.cxf.interceptors.EndpointCounterInterceptor</property-value>
    </property>
  </endpoint-config>
</jaxws-config>
```
Apache CXF HTTP Transport
In Apache CXF, HTTP transport configuration is achieved by specifying `org.apache.cxf.transport.http.HTTPConduit` options. JBossWS integration allows conduits to be modified programmatically using the Apache CXF API as follows.

You can also control and override the Apache CXF `HTTPConduit` default values by setting system properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cxf.client.allowChunking</td>
<td>Boolean</td>
<td>Specifies whether to send requests using chunking.</td>
</tr>
<tr>
<td>cxf.client.chunkingThreshold</td>
<td>Integer</td>
<td>Sets the threshold at which switching from non-chunking to chunking mode.</td>
</tr>
<tr>
<td>cxf.client.connectionTimeout</td>
<td>Long</td>
<td>Sets the number of milliseconds for the connection timeout.</td>
</tr>
<tr>
<td>cxf.client.receiveTimeout</td>
<td>Long</td>
<td>Sets the number of milliseconds for the receive timeout.</td>
</tr>
<tr>
<td>cxf.client.connection</td>
<td>String</td>
<td>Specifies whether to use the <code>Keep-Alive</code> or <code>close</code> connection type.</td>
</tr>
<tr>
<td>cxf.tls-client.disableCNCheck</td>
<td>Boolean</td>
<td>Specifies whether to disable the CN host name check.</td>
</tr>
</tbody>
</table>

5.1.3. WS-Security Changes
- If your application contains a custom callback handler that accesses the `org.apache.ws.security.WSPasswordCallback` class, be aware that this class has moved to package `org.apache.wss4j.common.ext`.

- Most of the SAML bean objects from the `org.apache.ws.security.saml.ext` package have been moved to the `org.apache.wss4j.common.saml` package.

- Use the RSA v1.5 key transport and all related algorithms are disallowed by default.

- The Security Token Service (STS) previously only validated `onBehalfOf` tokens. It now also validates `ActAs` tokens. As a consequence, a valid username and password must be specified in the `UsernameToken` that is provided for the `ActAs` token.

- SAML Bearer tokens are now required to have an internal signature. The `org.apache.wss4j.dom.validate.SamlAssertionValidator` class now has a `setRequireBearerSignature()` method to enable or disable the signature verification.

### 5.1.4. JBoss Modules Structure Change

The `cxf-api` and `cxf-rt-core` JARs have been merged into one `cxf-core` JAR. As a consequence, the `org.apache.cxf` module in JBoss EAP now contains the `cxf-core` JAR and exposes more classes than in the previous release.

### 5.1.5. Bouncy Castle Requirements and Changes

If you want to use AES encryption with Galois/Counter Mode (GCM) for symmetric encryption in XML/WS-Security, you need the BouncyCastle Security Provider.

JBoss EAP 7 ships with the `org.bouncycastle` module and JBossWS is now able to rely on its class loader to get and use the BouncyCastle Security Provider. Therefore it is no longer necessary to statically install BouncyCastle in the current JVM. For applications running outside of the container, the security provider can be made available to JBossWS by adding a BouncyCastle library to the class path.

You can disable this behavior by setting the `org.jboss.ws.cxf.noLocalBC` property value to `true` in the `jaxws-endpoint-config.xml` deployment descriptor file for the server or the `jaxws-client-config.xml` descriptor file for clients.

If you want to use a different version than the one that ships with JBoss EAP, you can still statically install BouncyCastle to the JVM. In that case, the statically installed BouncyCastle Security Provider is chosen over the provider present in the class path. To avoid any issues, you must use BouncyCastle 1.49, 1.51, or greater.

### 5.1.6. Apache CXF Bus Selection Strategy

The default bus selection strategy for clients running in-container has changed from `THREAD_BUS` to `TCCL_BUS`. For clients running out-of container, the default strategy is still `THREAD_BUS`. You can restore the behavior to that of the previous release by using either of the following methods.

- Boot the JBoss EAP server with the system property `org.jboss.ws.cxf.jaxws-client.bus.strategy` value set to `THREAD_BUS`.

- Explicitly set the selection strategy in the client code.
5.1.7. JAX-WS 2.2 Requirements for WebServiceRef

Containers must use JAX-WS 2.2 style constructors, which include the `WebServiceFeature` class as an argument in the constructor, to build clients that are injected into web service references. JBoss EAP 6.4, which ships with JBossWS 4, hides that requirement. JBoss EAP 7 ships with JBossWS 5, which no longer hides this requirement. This means that user provided service classes injected by the container must implement JAX-WS 2.2 or later by updating the existing code to use the `javax.xml.ws.Service` constructor that includes one or more `WebServiceFeature` arguments.

```java
protected Service(URL wsdlDocumentLocation,
                 QName serviceName,
                 WebServiceFeature... features)
```

5.1.8. IgnoreHttpsHost CN Check Change

In previous releases, you could disable the HTTPS URL hostname check against a service’s Common Name (CN) given in its certificate by setting the system property `org.jboss.security.ignoreHttpsHost` to `true`. This system property name has been replaced with `cxf.tls-client.disableCNCheck`.

5.1.9. Server Side Configuration and Class Loading

As a consequence of enabling injections into service endpoint and service client handlers, it is no longer possible to automatically load handler classes from the `org.jboss.as.webservices.server.integration` JBoss module. If your application depends on a given predefined configuration, you might need to explicitly define new module dependencies for your deployment. For more information, see `Migrate Explicit Module Dependencies`.

5.1.10. Deprecation of Java Endorsed Standards Override Mechanism

The `Java Endorsed Standards Override Mechanism` was deprecated in JDK 1.8_40 with intent to remove it in JDK 9. This mechanism allowed developers to make libraries available to all deployed applications by placing JARs into an endorsed directory within the JRE.

If your application uses the JBossWS implementation of Apache CXF, JBoss EAP 7 ensures the required dependencies are added in the correct order and you should not be impacted by this change. If your application accesses Apache CXF directly, you must now provide the Apache CXF dependencies after the JBossWS dependencies as part of your application deployment.

5.1.11. Specification of Descriptor in EAR Archive

In previous releases of JBoss EAP, you could configure the `jboss-webservices.xml` deployment descriptor file for EJB web service deployments in the `META-INF/` directory of JAR archives or in the `WEB-INF/` directory for POJO web service deployments and EJB web service endpoints bundled in WAR archives.

In JBoss EAP 7, you can now configure the `jboss-webservices.xml` deployment descriptor file in the `META-INF/` directory of an EAR archive. If a `jboss-webservices.xml` file is found both in the EAR archive and the JAR or WAR archive, the configuration data in the `jboss-webservices.xml` file for the JAR or WAR overrides the corresponding data in the EAR descriptor file.

5.2. UPDATE THE REMOTE URL CONNECTOR AND PORT

```java
protected
```
In JBoss EAP 7, the default connector has changed from remote to http-remoting and the default remote connection port has changed from 4447 to 8080. The JNDI provider URL for the default configuration has changed from remote://localhost:4447 to http-remoting://localhost:8080.

If you use the JBoss EAP 7 migrate operation to update your configuration, you do not need to modify the remote connector, remote port, or JNDI provider URLs because the migration operation preserves the JBoss EAP 6 remoting connector and 4447 port configuration settings in the subsystem configuration. For more information about the migrate operation, see Management CLI Migration Operation.

If you do not use the migrate operation and instead run with the new JBoss EAP 7 default configuration, you must change the remote connector, remote port, and JNDI provider URL to use the new settings as described above.

5.3. MESSAGING APPLICATION CHANGES

5.3.1. Replace or Update JMS Deployment Descriptors

The proprietary HornetQ JMS resource deployment descriptor files identified by the naming pattern -jms.xml no longer work in JBoss EAP 7. The following is an example of a JMS resource deployment descriptor file in JBoss EAP 6.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<messaging-deployment xmlns="urn:jboss:messaging-deployment:1.0">
  <hornetq-server>
    <jms-destinations>
      <jms-queue name="testQueue">
        <entry name="queue/test"/>
        <entry name="java:jboss/exported/jms/queue/test"/>
      </jms-queue>
      <jms-topic name="testTopic">
        <entry name="topic/test"/>
        <entry name="java:jboss/exported/jms/topic/test"/>
      </jms-topic>
    </jms-destinations>
  </hornetq-server>
</messaging-deployment>
```

If you used -jms.xml JMS deployment descriptors in your application in the previous release, you can either convert your application to use the standard Java EE 7 deployment descriptor as specified in section EE.5.18 of the Java EE 7 specification or you can update the deployment descriptor to use the messaging-activemq-deployment schema instead.

If you choose to update the descriptor, you need to make the following modifications.

- Change the namespace from "urn:jboss:messaging-deployment:1.0" to "urn:jboss:messaging-activemq-deployment:1.0".
- Change the <hornetq-server> element name to <server>.

The modified file should look like the following example.

```xml
<?xml version="1.0" encoding="UTF-8"?>
```
For information about server configuration changes related to messaging, see Messaging Server Configuration Changes.

5.3.2. Update External JMS Clients

JBoss EAP 7 still supports the JMS 1.1 API, so you do not need to modify your code.

The default remote connector and port has changed in JBoss EAP 7. For details about this change, see Update the Remote URL Connector and Port.

If you migrate your server configuration using the migrate operation, the old settings are preserved and you do not need to update your PROVIDER_URL. However, if you run with the new JBoss EAP 7 default configuration, you must change the PROVIDER_URL in the client code to use the new http-remoting://localhost:8080 setting. For more information, see Migrate Remote Naming Client Code.

If you plan to migrate your code to use the JMS 2.0 API, see the helloworld-jms quickstart for a working example.

5.3.3. Replace the HornetQ API

JBoss EAP 6 included the org.hornetq module, which allowed you to use the HornetQ API in your application source code.

Apache ActiveMQ Artemis replaces HornetQ in JBoss EAP 7, so you must migrate any code that used the HornetQ API to use the Apache ActiveMQ Artemis API. The libraries for this API are included in the org.apache.activemq.artemis module.

ActiveMQ Artemis is an evolution of HornetQ, so many of the concepts still apply.

5.4. JAX-RS AND RESTEASY APPLICATION CHANGES

The previous release of JBoss EAP bundled RESTEasy 2, which was an implementation of JAX-RS 1.x. JBoss EAP 7 includes RESTEasy 3, which is an implementation of JAX-RS 2.0 as defined by the JSR 339: JAX-RS 2.0: The Java API for RESTful Web Services specification. For more information about the Java API for RESTful Web Services, see the JAX-RS 2.0 API Specification.

The version of Jackson included in JBoss EAP has changed. The previous version of JBoss EAP included Jackson 1.9.9. JBoss EAP 7 now includes Jackson 2.6.3 or greater.
This section describes how these changes might impact applications that use RESTEasy or JAX-RS.

## 5.4.1. RESTEasy Deprecated Classes

### Interceptor and MessageBody Classes

JSR 311: JAX-RS: The Java™ API for RESTful Web Services did not include an interceptor framework, so RESTEasy 2 provided one. JSR 339: JAX-RS 2.0: The Java API for RESTful Web Services introduced an official interceptor and filter framework, so the interceptor framework included in RESTEasy 2 is now deprecated, and is replaced by the JAX-RS 2.0 compliant interceptor facility in RESTEasy 3.x. The relevant interfaces are defined in the `javax.ws.rs.ext` package of the `jaxrs-api` module.

- The following interceptor interfaces are deprecated in RESTEasy 3.x.
  - `org.jboss.resteasy.spi.interception.PreProcessInterceptor`
  - `org.jboss.resteasy.spi.interception.PostProcessInterceptor`
  - `org.jboss.resteasy.spi.interception.ClientExecutionInterceptor`
  - `org.jboss.resteasy.spi.interception.ClientExecutionContext`
  - `org.jboss.resteasy.spi.interception.AcceptedByMethod`

- The `org.jboss.resteasy.spi.interception.PreProcessInterceptor` interface is replaced by the `javax.ws.rs.container.ContainerRequestFilter` interface in RESTEasy 3.x.

- The following interfaces and classes are also deprecated in RESTEasy 3.x.
  - `org.jboss.resteasy.spi.interception.MessageBodyReaderInterceptor`
  - `org.jboss.resteasy.spi.interception.MessageBodyWriterInterceptor`
  - `org.jboss.resteasy.spi.interception.MessageBodyWriterContext`
  - `org.jboss.resteasy.spi.interception.MessageBodyReaderContext`
  - `org.jboss.resteasy.core.interception.InterceptorRegistry`
  - `org.jboss.resteasy.core.interception.InterceptorRegistryListener`
  - `org.jboss.resteasy.core.interception.ClientExecutionContextImpl`

- The `org.jboss.resteasy.spi.interception.MessageBodyWriterInterceptor` interface is replaced by the `javax.ws.rs.ext.WriterInterceptor` interface.

- In addition, some changes to the `javax.ws.rs.ext.MessageBodyWriter` interface might not be backward compatible with respect to JAX-RS 1.x. If your application used JAX-RS 1.x, review your application code to make sure you define `@Produces` or `@Consumes` for your endpoints. Failure to do so might result in an error similar to the following.

```
org.jboss.resteasy.core.NoMessageBodyWriterFoundFailure: Could not find MessageBodyWriter for response object of type: <OBJECT> of media type:
```

The following is an example of a REST endpoint that can cause this error.
To fix the issue, add the import for `javax.ws.rs.Produces` and the `@Produces` annotation as follows.

```java
import javax.ws.rs.Produces;
...
@Path("dates")
public class DateService {
    @GET
    @Path("daysuntil/{targetdate}")
    @Produces(MediaType.TEXT_PLAIN)
    public long showDaysUntil(@PathParam("targetdate") String targetDate) {
        DateLogger.LOGGER.logDaysUntilRequest(targetDate);
        final long days;
        try {
            final LocalDate date = LocalDate.parse(targetDate, DateTimeFormatter.ISO_DATE);
            days = ChronoUnit.DAYS.between(LocalDate.now(), date);
        } catch (DateTimeParseException ex) {
            // ** DISCLAIMER **. This example is contrived.
            throw new WebApplicationException(Response.status(400).entity(ex.getLocalizedMessage()).type(MediaType.TEXT_PLAIN).build());
        }
        return days;
    }
}
```

To fix the issue, add the import for `javax.ws.rs.Produces` and the `@Produces` annotation as follows.
NOTE

All interceptors from the previous release of RESTEasy can run in parallel with the new JAX-RS 2.0 filter and interceptor interfaces.

For more information about interceptors, see RESTEasy Interceptors in Developing Web Services Applications for JBoss EAP.

For more information about the new replacement API, see the RESTEasy JAX-RS 3.0.13.Final API or later.

Client API
The RESTEasy client framework in resteasy-jaxrs has been replaced by the JAX-RS 2.0 compliant resteasy-client module. As a result, some RESTEasy client API classes and methods are deprecated.

- The following classes are deprecated.
  - org.jboss.resteasy.client.ClientRequest
  - org.jboss.resteasy.client.ClientRequestFactory
  - org.jboss.resteasy.client.ClientResponse
  - org.jboss.resteasy.client.ProxyBuilder
  - org.jboss.resteasy.client.ProxyConfig
  - org.jboss.resteasy.client.ProxyFactory

- The org.jboss.resteasy.client.ClientResponseFailure exception and the org.jboss.resteasy.client.ClientExecutor and org.jboss.resteasy.client.EntityTypeFactory interfaces are also deprecated.


The following is an example of how to send a link header with the RESTEasy client in RESTEasy 2.3.x.

```java
ClientRequest request = new
ClientRequest(generateURL("/linkheader/str"));
request.addLink("previous chapter", "previous",
"http://example.com/TheBook/chapter2", null);
ClientResponse response = request.post();
LinkHeader header = response.getLinkHeader();
```

The following is an example of how to accomplish the same task with the RESTEasy client in RESTEasy 3.

```java
return days;
```
See the resteasy-jaxrs-client quickstart for an example of an external JAX-RS RESTEasy client that interacts with a JAX-RS Web service.

- The classes and interfaces in the `org.jboss.resteasy.client.cache` package are also deprecated. They are replaced by equivalent classes and interfaces in the `org.jboss.resteasy.client.jaxrs.cache` package.

**NOTE**

For more information about the `org.jboss.resteasy.client.jaxrs` API classes, see the RESTEasy JAX-RS JavaDoc.

StringConverter

The `org.jboss.resteasy.spi.StringConverter` class is deprecated in RESTEasy 3.x. This functionality can be replaced using the JAX-RS 2.0 `jakarta.ws.rs.ext.ParamConverterProvider` class.

### 5.4.2. Removed or Protected RESTEasy Classes

**ResteasyProviderFactory Add methods**

Most of the `org.jboss.resteasy.spi.ResteasyProviderFactory` add() methods have been removed or made protected in RESTEasy 3.0. For example, the `addBuiltInMessageBodyReader()` and `addBuiltInMessageBodyWriter()` methods have been removed and the `addMessageBodyReader()` and `addMessageBodyWriter()` methods have been made protected.

You should now use the `registerProvider()` and `registerProviderInstance()` methods.

**Additional Classes Removed From RESTEasy 3**

The `@org.jboss.resteasy.annotations.cache.ServerCached` annotation, which specified the response to the JAX-RS method should be cached on the server, was removed from RESTEasy 3 and must be removed from the application code.

### 5.4.3. Additional RESTEasy Changes

**SignedInput and SignedOuput**

- SignedInput and SignedOutput for resteasy-crypto must have the `Content-Type` set to `multipart/signed` in either the Request or Response object, or by using the `@Consumes` or `@Produces` annotation.

- SignedOutput and SignedInput can be used to return the `application/pkcs7-signature` MIME type format in binary form by setting that type in the `@Produces` or `@Consumes` annotations.

- If the `@Produces` or `@Consumes` is `text/plain` MIME type, SignedOutput will be base64 encoded and sent as a String.
Security Filters
The security filters for @RolesAllowed, @PermitAll, and @DenyAll now return "403 Forbidden" instead of "401 Unauthorized".

Client-side Filters
The new JAX-RS 2.0 client-side filters will not be bound and run when you are using the RESTEasy client API from the previous release.

Asynchronous HTTP Support
Because the JAX-RS 2.0 specification adds asynchronous HTTP support using the @Suspended annotation and the AsynResponse interface, the RESTEasy proprietary API for asynchronous HTTP has been deprecated and might be removed as soon as RESTEasy 3.1. The asynchronous Tomcat and asynchronous JBoss Web modules have also been removed from the server installation. If you are not using the Servlet 3.0 container or higher, asynchronous HTTP server-side processing will be simulated and run synchronously in same request thread.

Server-side Cache
Server-side cache setup has changed. Please see the RESTEasy Documentation for more information.

YAML Provider Setting Changes
In previous releases of JBoss EAP, the RESTEasy YAML provider setting was enabled by default. This has changed in JBoss EAP 7. The YAML provider is now disabled by default. Its use is not supported due to a security issue in the SnakeYAML library used by RESTEasy for unmarshalling and it must be explicitly enabled in the application. For information about how to enable the YAML provider in your application and add the Maven dependencies, see YAML Provider in Developing Web Services Applications for JBoss EAP.

SerializableProvider
Deserializing Java objects from untrusted sources is not safe. For this reason, in JBoss EAP 7, the org.jboss.resteasy.plugins.providers.SerializableProvider class is disabled by default, and it is not recommended to use this provider.

5.4.4. RESTEasy SPI Changes

SPI Exceptions
All SPI failure exceptions have been deprecated and are no longer used internally. They have been replaced with the corresponding JAX-RS 2.0 exception.

<table>
<thead>
<tr>
<th>Deprecated Exception</th>
<th>Replacement Exception in jaxrs-api module</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.jboss.resteasy.spi.ForbiddenException</td>
<td>javax.ws.rs.ForbiddenException</td>
</tr>
<tr>
<td>org.jboss.resteasy.spi.MethodNotAllowedException</td>
<td>javax.ws.rs.NotAllowedException</td>
</tr>
<tr>
<td>org.jboss.resteasy.spi.NotAcceptableException</td>
<td>javax.ws.rs.NotAcceptableException</td>
</tr>
<tr>
<td>org.jboss.resteasy.spi.NotFoundException</td>
<td>javax.ws.rs.NotFoundException</td>
</tr>
<tr>
<td>org.jboss.resteasy.spi.UnauthorizedException</td>
<td>javax.ws.rs.NotAuthorizedException</td>
</tr>
<tr>
<td>org.jboss.resteasy.spi.UnsupportedMediaTypeException</td>
<td>javax.ws.rs.NotSupportedException</td>
</tr>
</tbody>
</table>
InjectorFactory and Registry
The InjectorFactory and Registry SPIs have changed. This should not be an issue if you use RESTEasy as documented and supported.

5.4.5. Jackson Provider Changes
The version of Jackson included in JBoss EAP has changed. The previous version of JBoss EAP included Jackson 1.9.9. JBoss EAP 7 now includes Jackson 2.6.3 or greater. As a result, the Jackson provider has changed from resteasy-jackson-provider to resteasy-jackson2-provider.

The upgrade to the resteasy-jackson2-provider requires some package changes. For example, the Jackson annotation package has changed from org.codehaus.jackson.annotate to com.fasterxml.jackson.annotation.

To switch your application to use the default provider that was included in the previous release of JBoss EAP, see Switching the Default Jackson Provider in Developing Web Services Applications for JBoss EAP.

5.4.6. Spring RESTEasy Integration Changes
The Spring 4.0 framework introduced support for Java 8. If you plan to use the RESTEasy 3.x integration with Spring, be sure to specify 4.2.x as the minimum Spring version in your deployment as this is the earliest stable version supported by JBoss EAP 7.

5.4.7. RESTEasy Jettison JSON Provider Changes
The RESTEasy Jettison JSON provider is deprecated in JBoss EAP 7 and is no longer added to deployments by default. You are encouraged to switch to the recommended RESTEasy Jackson provider. If you prefer to continue to use the Jettison provider, you must define an explicit dependency for it in the jboss-deployment-descriptor.xml file as demonstrated in the following example.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jboss-deployment-structure>
  <deployment>
    <exclusions>
      <module name="org.jboss.resteasy.resteasy-jackson2-provider"/>
      <module name="org.jboss.resteasy.resteasy-jackson-provider"/>
    </exclusions>
    <dependencies>
      <module name="org.jboss.resteasy.resteasy-jettison-provider" services="import"/>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

For more information about how to define explicit dependencies, see Add an Explicit Module Dependency to a Deployment in the JBoss EAP Development Guide.

5.5. CDI 1.2 APPLICATION CHANGES
JBoss EAP 7 includes support for CDI 1.2. As a result, applications written using CDI 1.0 might see some changes in behavior when migrated to JBoss EAP 7. This section summarizes only a few of those changes.
You can find more information about Weld and CDI 1.2 in the following references:

- Contexts and Dependency Injection for the Java EE platform
- CDI 1.2 Javadoc
- Weld 2.3.3.Final - CDI Reference Implementation

**Bean Archives**

Bean classes of enabled beans must be deployed in bean archives to ensure they are scanned by CDI to find and process the bean classes.

In CDI 1.0, an archive was defined as an *explicit* bean archive if it contained a `beans.xml` file in the `META-INF/` directory for an application client, EJB, or library JAR, or if it contained a `beans.xml` file in the `WEB-INF/` directory for a WAR.

CDI 1.1 introduced *implicit* bean archives, which are archives that contain one or more bean classes with a bean defining annotation, or one or more session beans. Implicit bean archives are scanned by CDI and, during type discovery, only classes with bean defining annotations are discovered. For more information, see Type and Bean Discovery in *Contexts and Dependency Injection for the Java EE platform*.

A bean archive has a bean discovery mode of *all*, *annotated* or *none*. A bean archive that contains a `beans.xml` file with no version has a default bean discovery mode of *all*. A bean archive that contains a `beans.xml` file with version 1.1 or later must specify the `bean-discovery-mode` attribute. The default value for the attribute is *annotated*.

An archive is not a bean archive in the following cases:

- It contains a `beans.xml` file with a `bean-discovery-mode` of *none*.
- It contains a CDI extension with no `beans.xml` file.

An archive is an *explicit* bean archive in the following cases:

- The archive contains a `beans.xml` file with a version number of 1.1 or later and a `bean-discovery-mode` of *all*.
- The archive contains a `beans.xml` file with no version number.
- The archive contains an empty `beans.xml` file.

An archive is an *implicit* bean archive in the following cases:

- The archive contains one or more bean classes with a bean defining annotation, or one or more session beans, even if it does not contain a `beans.xml` file.
- The archive contains a `beans.xml` file with a `bean-discovery-mode` of *annotated*.

CDI 1.2 limited bean defining annotations to the following:

- `@ApplicationScoped`, `@SessionScoped`, `@ConversationScoped`, and `@RequestScoped` annotations
- All other normal scope types
- `@Interceptor` and `@Decorator` annotations
- All stereotype annotations, which are annotations annotated with `@Stereotype`
- `@Dependent` scope annotation

For more information about bean archives, see Bean Archives in Contexts and Dependency Injection for the Java EE platform.

**Clarification of Conversation Resolution**
The conversation context lifecycle was changed to prevent conflicts with the Servlet specification as described in CDI Specification Issue CDI-411. The conversation scope is active during all servlet requests and should not prevent other servlets or servlet filters from setting the request body or character encoding.

**Observer Resolution**
Event resolution has been partly rewritten in CDI 1.2. In CDI 1.0, an event is delivered to an observer method if the observer method has all the event qualifiers. In CDI 1.2, an event is delivered to an observer method if the observer method has no event qualifiers or has a subset of the event qualifiers.

### 5.6. MIGRATE EXPLICIT MODULE DEPENDENCIES

The introduction of the modular class loading system and JBoss Modules in the previous release of JBoss EAP allowed for fine-grained control of the classes available to applications. This feature allowed you to configure explicit module dependencies using the application's `MANIFEST.MF` file or the `jboss-deployment-structure.xml` deployment descriptor file.

If you defined explicit module dependencies in your application, you should be aware of the following changes in JBoss EAP 7.

**Review Dependencies for Availability**
The modules that are included in JBoss EAP have changed. When you migrate your application to JBoss EAP 7, review your `MANIFEST.MF` and `jboss-deployment-structure.xml` file entries to make sure they do not refer to any modules that were removed from this release of the product.

**Dependencies That Require Annotation Scanning**
In the previous release of JBoss EAP, if your dependency contained annotations that needed to be processed during annotation scanning, such as when declaring EJB Interceptors, you were required to generate and include a Jandex index in a new JAR file and then set a flag in the `MANIFEST.MF` or `jboss-deployment-structure.xml` deployment descriptor file.

JBoss EAP 7 now provides automatic runtime generation of annotation indexes for static modules, so you no longer need to generate them manually. However, you still need to add the `annotations` flag to the application's `MANIFEST.MF` file or the `jboss-deployment-structure.xml` deployment descriptor file as demonstrated below.

---

**Example Annotation Flag in the MANIFEST.MF File**
```
Dependencies: com.company.my-ejb annotations, com.company.other
```

**Example Annotation Flag in the jboss-deployment-structure.xml File**
```
<jboss-deployment-structure>
<deployment>
<dependencies>
  <module name="com.company.my-ejb" annotations="true"/>
</dependencies>
</deployment>
</jboss-deployment-structure>
```
5.7. HIBERNATE AND JPA MIGRATION CHANGES

5.7.1. Hibernate ORM 3.0

The integration classes that made it easier to use Hibernate ORM 3 in the previous release were removed from JBoss EAP 7. If your application still uses Hibernate ORM 3 libraries, it is strongly recommended that you migrate your application to use Hibernate ORM 5 as Hibernate ORM 3 will no longer work in JBoss EAP without a lot of effort. If you can not migrate to Hibernate ORM 5, you must define a custom JBoss Module for the Hibernate ORM 3 JARs and exclude the Hibernate ORM 5 classes from your application.

5.7.2. Hibernate ORM 4.0 - 4.3

If your application needs second-level cache enabled, you should migrate to Hibernate ORM 5, which is integrated with Infinispan 8.x.

Applications written with Hibernate ORM 4.x can still use Hibernate ORM 4.x. You must define a custom JBoss module for the Hibernate ORM 4.x JARs and exclude the Hibernate ORM 5 classes from your application. However, it is strongly recommended that you rewrite your application code to use Hibernate ORM 5. For information about migrating to Hibernate ORM 5, see Migrating to Hibernate ORM 5.

5.7.3. Hibernate ORM 5

If your application contains a persistence.xml file or the code uses the @PersistenceContext or @PersistenceUnit annotations, JBoss EAP 7 detects this during deployment and assumes the application uses JPA. It implicitly adds the Hibernate ORM 5 libraries plus a few other dependencies to your application class path and defaults to using these libraries.

5.7.4. Migrating to Hibernate ORM 5

This section highlights the changes you need to make when migrating from Hibernate ORM version 4.3 to version 5. For more information about the changes implemented between Hibernate ORM 4 and Hibernate ORM 5, see the Hibernate ORM 5.0 Migration Guide.

Removed and Deprecated Classes

The following deprecated classes were removed from Hibernate ORM 5.

- org.hibernate.cfg.AnnotationConfiguration
- org.hibernate.id.TableGenerator
- org.hibernate.id.TableHiLoGenerator
- org.hibernate.id.SequenceGenerator

Other Changes to Classes and Packages
The `org.hibernate.integrator.spi.Integrator` interface changed to account for bootstrap redesign.

A new package `org.hibernate.engine.jdbc.env` package was created. It contains the `org.hibernate.engine.jdbc.env.spi.JdbcEnvironment` interface, which was extracted from `org.hibernate.engine.jdbc.spi.JdbcServices` interface.

A new `org.hibernate.boot.model.relational.ExportableProducer` interface was introduced that will affect `org.hibernate.id.PersistentIdentifierGenerator` implementations.

The signature of `org.hibernate.id.Configurable` was changed to accept `org.hibernate.service.ServiceRegistry` rather than just `org.hibernate.dialect.Dialect`.

The `org.hibernate.metamodel.spi.TypeContributor` interface has migrated to `org.hibernate.boot.model.TypeContributor`.

The `org.hibernate.metamodel.spi.TypeContributions` interface has migrated to `org.hibernate.boot.model.TypeContributions`.

**Type Handling**

- Built-in `org.hibernate.type.descriptor.sql.SqlTypeDescriptor` implementations no longer auto-register themselves with `org.hibernate.type.descriptor.sql.SqlTypeDescriptorRegistry`. Applications using custom `SqlTypeDescriptor` implementations that extend the built-in implementations and rely on that behavior must be updated to call `SqlTypeDescriptorRegistry.addDescriptor()` themselves.

- For IDs defined as generated UUIDs, some databases require you to explicitly set the `@Column(length=16)` in order to generate `BINARY(16)` so that comparisons work properly.

- For `EnumType` mappings defined in the `hbm.xml`, where you want `javax.persistence.EnumType.STRING` name-mapping, this configuration must be explicitly stated by using either the `useNamed(true)` setting or by specifying a `VARCHAR` value of 12.

**Transaction Management**

- The transaction SPI underwent a major redesign in Hibernate ORM 5. In Hibernate ORM 4.3, you used the `org.hibernate.Transaction` API to directly access different back-end transaction strategies. Hibernate ORM 5 introduced a level of indirection. On the back end, the `org.hibernate.Transaction` implementation now talks to a `org.hibernate.resource.transaction.TransactionCoordinator`, which represents the transactional context for a given session according to the back-end strategy. While this does not have a direct impact on developers, it could affect the bootstrap configuration. Previously applications would specify `hibernate.transaction.factory_class` property, which is now deprecated, and refer to a `org.hibernate.engine.transaction.spi.TransactionFactory` FQN (fully qualified name). With Hibernate ORM 5, you specify the `hibernate.transaction.coordinator_class` setting and refer to a `org.hibernate.resource.transaction.TransactionCoordinatorBuilder`. See `org.hibernate.cfg.AvailableSettings.TRANSACTION_COORDINATOR_STRATEGY` for additional details.
The following short names are now recognized.

- **jdbc**: Manage transactions using the JDBC `java.sql.Connection`. This is the default for non-JPA transactions.
- **jta**: Manage transactions using JTA.

**IMPORTANT**

If a JPA application does not provide a setting for the `hibernate.transaction.coordinator_class` property, Hibernate will automatically build the proper transaction coordinator based on the transaction type for the persistence unit.

If a non-JPA application does not provide a setting for the `hibernate.transaction.coordinator_class` property, Hibernate will default to `jdbc` to manage the transactions. This default will cause problems if the application actually uses JTA-based transactions. A non-JPA application that uses JTA-based transactions should explicitly set the `hibernate.transaction.coordinator_class` property value to `jta` or provide a custom `org.hibernate.resource.transaction.TransactionCoordinatorBuilder` that builds a `org.hibernate.resource.transaction.TransactionCoordinator` that properly coordinates with JTA-based transactions.

**Other Hibernate ORM 5 Changes**

- The `cfg.xml` files are again fully parsed and integrated with events, security, and other functions.

- The properties loaded from the `cfg.xml` using the `EntityManagerFactory` did not previously prefix names with `hibernate`. This has now been made consistent.

- The configuration is no longer serializable.

- The `org.hibernate.dialect.Dialect.getQuerySequencesString()` method now retrieves catalog, schema, and increment values.

- The `AuditConfiguration` modifier was removed from `org.hibernate.envers.boot.internal.EnversService`.

- The `AuditStrategy` method parameters were changed to remove the obsolete `AuditConfiguration` and use the new `EnversService`.

- Various classes and interfaces in the `org.hibernate.hql.spi` package and subpackages have been moved to the new `org.hibernate.hql.spi.id` package. This includes the `MultiTableBulkIdStrategy` class and the `AbstractTableBasedBulkIdHandler`, `TableBasedDeleteHandlerImpl`, and `TableBasedUpdateHandlerImpl` interfaces and their subclasses.

- There was a complete redesign of property access contracts.

- Valid `hibernate.cache.default_cache_concurrency_strategy` setting values are now defined using the
org.hibernate.cache.spi.access.AccessType.getExternalName() method rather than the org.hibernate.cache.spi.access.AccessType enum constants. This is more consistent with other Hibernate settings.

5.8. HIBERNATE SEARCH CHANGES

The version of Hibernate Search that ships with JBoss EAP 7 has changed. The previous release of JBoss EAP shipped with Hibernate Search 4.6.x. JBoss EAP 7 ships with Hibernate Search 5.5.x.

Hibernate Search 5.5 is built upon Apache Lucene 5.3.1. If you use any native Lucene APIs, be sure to align with this version. The Hibernate Search 5.5 API wraps and hides the complexity of many of the Lucene API changes made between version 3 and version 5, however, some classes are now deprecated, renamed, or repackaged. This section describes how these changes might impact your application code.

**Hibernate Search Mapping Changes**

**Indexing of id Fields of Embedded Relations**

When using an @IndexedEmbedded annotation to include fields from a related entity, the id of the related entity is no longer included. You can enable the inclusion of the id by using the includeEmbeddedObjectId attribute of the @IndexedEmbedded annotation.

```java
@IndexedEmbedded(includeEmbeddedObjectId=true)
```

**Number and Date Index Formatting Changes**

Numbers and dates are now indexed as numeric fields by default. Properties of type int, long, float, double, and their corresponding wrapper classes are no longer indexed as strings. Instead, they are now indexed using Lucene’s appropriate numeric encoding. The id fields are an exception to this rule. Even when they are represented by a numeric type, they are still indexed as a string by default. The use of @NumericField is now obsolete unless you want to specify a custom precision for the numeric encoding. You can keep the old string-based index format by explicitly specifying a string encoding field bridge. In the case of integers, this is the org.hibernate.search.bridge.builtin.IntegerBridge. Check the org.hibernate.search.bridge.builtin package for other publicly available field bridges.

**Date** and **Calendar** are no longer indexed as strings. Instead, instances are encoded as long values representing the number of milliseconds since January 1, 1970, 00:00:00 GMT. You can switch the indexing format by using the new EncodingType enum. For example:

```java
@DateBridge(encoding=EncodingType.STRING)
@CalendarBridge(encoding=EncodingType.STRING)
```

The encoding change for numbers and dates is important and can have a big impact on application behavior. If you have a query that targets a field that was previously string-encoded, but is now encoded numerically, you must update the query. Numeric fields must be searched with a NumericRangeQuery. You must also make sure that all fields targeted by faceting are string encoded. If you use the Search query DSL, the correct query should be created automatically for you.

**Miscellaneous Hibernate Search Changes**

- Sorting options have improved and field encoding specified incorrectly for sorting options now results in runtime exceptions. Lucene also offers more performant sorting if the fields used in the sort are known up front. Hibernate Search 5.5 provides the new @SortableField annotation and its multi-valued companion @SortableFields. See the Migration Guide from Hibernate Search 5.4 to 5.5 for more information.
- The Lucene `SortField` API requires the following application code change. In the previous release of JBoss EAP, you set the type of the sort field in the query as follows.

```java
fulltextQuery.setSort(new Sort(new SortField("title", SortField.STRING)));
```

The following is an example of how you set it in JBoss EAP 7.

```java
fulltextQuery.setSort(new Sort(new SortField("title", SortField.Type.STRING)));
```

- Since `SearchFactory` should only be used by ORM integration, it was moved from the `hibernate-search-engine` module to the `hibernate-search-orm` module. Other integrators should depend exclusively on `SearchIntegrator`, which replaces the deprecated `SearchFactoryIntegrator`.

- The enum value `SpatialMode.GRID` was renamed to `SpatialMode.HASH`.

- `FullTextIndexEventListener` is now a final class. If you currently extend this class, you must find an alternate solution to achieve the same functionality.

- The `hibernate-search-analyzers` module was removed. The recommended approach is to directly use the appropriate Lucene artifact, for example, `org.apache.lucene:lucene-analyzers-common`.

- The JMS controller API has changed. The JMS back-end dependency on Hibernate ORM was removed so that it could be used in other non-ORM environments. A consequence is that implementors of `org.hibernate.search.backend.impl.jms.AbstractJMSHibernateSearchController` must adjust to the new signature. This class is an internal class and it is recommended to use it as an example instead of extending it.

- The `org.hibernate.search.spi.ServiceProvider` SPI was refactored. If you were integrating with the old service contract, refer to the Hibernate Search 5.5 Javadoc of `ServiceManager`, `Service`, `Startable` and `Stoppable` for details about the new contract.

- If you have kept indexes generated by Lucene 3.x and have not rebuilt them with Hibernate Search 5.0 or later, you will get an `IndexFormatTooOldException`. It is recommended that you rebuild the indexes with the mass indexer. If you are not able to do that, try to use Lucene's `IndexUpgrader`. You must carefully update the Hibernate Search mappings in case the default behavior has changed. For more information, see the Apache Lucene Migration Guide.

- Apache Lucene was upgraded from 3.6 to 5.3 in JBoss EAP 7. If your code imports Lucene code directly, see the Apache Lucene Migration Guide for details of the changes. Additional information can also be found in the Lucene Change Log.

- When using `@Field(indexNullAs=)` to encode a null marker value in the index, the type of the marker must be compatible with all other values that are indexed in that same field. For example, it was previously possible to encode a null value for numeric fields using a string "null". This is no longer allowed. Instead, you must choose a number to represent the null value, such as -1.
Significant improvements were made to the faceting engine. Most of the changes do not affect the API. The one notable exception is that you must now annotate any fields you intend to use for faceting with the `@Facet` or `@Facets` annotation.

**Hibernate Search Renamed and Repackaged Classes**
The following is a list of Hibernate Search classes that were repackaged or renamed.

<table>
<thead>
<tr>
<th>Previous Package and Class</th>
<th>New Package and Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.hibernate.search.SearchException</td>
<td>org.hibernate.search.exception.SearchException</td>
</tr>
</tbody>
</table>

**Lucene - Renamed and Repackaged Classes**
Query parsers were moved to a new module, resulting in a packaging change from `org.apache.lucene.queryParser.QueryParser` to `org.apache.lucene.queryparser.classic.QueryParser`.

Many of the Lucene analyzers were refactored, resulting in packaging changes. See the [Apache Lucene Documentation](#) to find the replacement packages.

Some Apache Solr utility classes, for example `TokenizerFactory` or `TokenFilterFactory`, were moved into Apache Lucene. If your application uses those utilities or custom analyzers, you must find the new package name in Apache Lucene.

See the [Apache Lucene Migration Guide](#) for more information.

**Hibernate Search Deprecated APIs**
For the complete list of Hibernate Search deprecated interfaces, classes, enums, annotation types, methods, constructors, and enum constants, see the [Hibernate Search Deprecated API](#) document.

**Hibernate Search Deprecated Interfaces**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.hibernate.search.store.Workspace</td>
<td>This interface will be moved and should be considered non-public API. For more information, see <code>HSEARCH-1915</code>.</td>
</tr>
</tbody>
</table>

**Hibernate Search Deprecated Classes**
<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.hibernate.search.filter.FilterKey</td>
<td>Custom filter keys are deprecated and are scheduled for removal in Hibernate Search 6. As of Hibernate Search 5.1, keys for caching Lucene filters are calculated automatically based on the given filter parameters.</td>
</tr>
<tr>
<td>org.hibernate.search.filter.StandardFilterKey</td>
<td>Custom filter keys are deprecated and are scheduled for removal in Hibernate Search 6. As of Hibernate Search 5.1, keys for caching Lucene filters are calculated automatically based on the given filter parameters.</td>
</tr>
</tbody>
</table>

### Hibernate Search Deprecated Enums

<table>
<thead>
<tr>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.hibernate.search.annotations.FieldCacheType</td>
<td>Remove the <code>CacheFromIndex</code> annotation as it is deprecated. See <a href="#">Hibernate Search Deprecated Annotations</a>.</td>
</tr>
</tbody>
</table>

### Hibernate Search Deprecated Annotations

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.hibernate.search.annotations.CacheFromIndex</td>
<td>Remove the annotation. No alternative replacement is necessary.</td>
</tr>
<tr>
<td>org.hibernate.search.annotations.Key</td>
<td>Custom filter cache keys are a deprecated feature and are scheduled to be removed in Hibernate Search 6. As of Hibernate Search 5.1, the filter cache keys are determined automatically based on the filter parameters so it is no longer required to provide a key object.</td>
</tr>
</tbody>
</table>

### Hibernate Search Deprecated Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.hibernate.search.FullTextSharedSessionBuilder.autoClose()</td>
<td>No replacement</td>
</tr>
<tr>
<td>org.hibernate.search.FullTextSharedSessionBuilder.autoClose(boolean)</td>
<td>No replacement</td>
</tr>
<tr>
<td>org.hibernate.search.cfg.IndexedMapping.cacheFromIndex(FieldCacheType...)</td>
<td>This will be removed with no replacement.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>org.hibernate.search.cfg.EntityDescriptor.getCacheInMemory()</td>
<td>This will be removed with no replacement.</td>
</tr>
<tr>
<td>org.hibernate.search.cfg.ContainedInMapping.numericField()</td>
<td>Invoke <code>field().numericField()</code> instead.</td>
</tr>
<tr>
<td>org.hibernate.search.cfg.EntityDescriptor.setCacheInMemory(Map&lt;String, Object&gt;)</td>
<td>This will be removed with no replacement.</td>
</tr>
<tr>
<td>org.hibernate.search.MassIndexer.threadsForSubsequentFetching(int)</td>
<td>This method will be removed.</td>
</tr>
<tr>
<td>org.hibernate.search.query.dsl.FuzzyContext.withThreshold(float)</td>
<td>Use <code>FuzzyContext.withEditDistanceUpTo(int)</code></td>
</tr>
</tbody>
</table>

### Hibernate Search Deprecated Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
</table>

### Changes Impacting Advanced Integrators

This section describes changes that are not part of the public API. They should not impact the average developer as these artifacts should only be accessed by integrators who extend the Hibernate Search framework.

- The `IndexWriterSetting.MAX_THREAD_STATES` and `IndexWriterSetting.TERM_INDEX_INTERVAL` enum constants are deprecated. They affect which properties are read from the configuration, so the fact they are missing means that configuration properties such as `hibernate.search.Animals.2.indexwriter.term_index_interval = default` are now ignored. The only side effect is that the property is not applied.

- The `SearchFactoryIntegrator` interface is now deprecated. You should immediately migrate all code to use `SearchIntegrator`.

- The `SearchFactoryBuilder` class is now deprecated. Use `SearchIntegrationBuilder` instead.

- The `HSQuery.getExtendedSearchIntegrator()` method has been deprecated. It might be possible to use `SearchIntegrator`, but it is preferable to remove it altogether.
- The `DocumentBuilderIndexedEntity.getFieldCacheOption()` method has been deprecated. There is no replacement.

- The `BuildContext.getIndexingStrategy()` method is deprecated. Use `BuildContext.getIndexingMode()` instead.


- The following is a list of Hibernate Search classes that were repackaged or renamed.

<table>
<thead>
<tr>
<th>Previous Package and Class</th>
<th>New Package and Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.hibernate.search.indexes.impl.DirectoryBasedIndexManager</td>
<td>org.hibernate.search.indexes.spi.DirectoryBaseIndexManager</td>
</tr>
<tr>
<td>org.hibernate.search.spi.SearchFactoryBuilder</td>
<td>org.hibernate.search.spi.SearchIntegratorBuilder</td>
</tr>
<tr>
<td>org.hibernate.search.spi.SearchFactoryIntegrator</td>
<td>org.hibernate.search.spi.SearchIntegrator</td>
</tr>
</tbody>
</table>

5.9. MIGRATE ENTITY BEANS TO JPA

Support for EJB Entity Beans is optional in Java EE 7 and they are not supported in JBoss EAP 7. This means container-managed persistence (CMP) and bean-managed persistence (BMP) entity beans must be rewritten to use Java Persistence API (JPA) entities when you migrate to JBoss EAP 7.

In previous releases of JBoss EAP, entity beans were created in application source code by extending the `javax.ejb.EntityBean` class and implementing the required methods. They were then configured in the `ejb-jar.xml` file. A CMP entity bean was specified using an `<entity>` element that contained a `<persistence-type>` child element with a value of `Container`. A BMP entity bean was specified using an `<entity>` element that contained a `<persistence-type>` child element with a value of `Bean`.

In JBoss EAP 7, you must replace any CMP and BMP entity beans in your code with Java Persistence API (JPA) entities. JPA entities are created using the `javax.persistence.*` classes and are defined in the `persistence.xml` file.

The following is an example of a JPA entity class.

```java
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
```
The following is an example of a persistence.xml file.

```xml
<persistence version="2.1"
    xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://xmlns.jcp.org/xml/ns/persistence
        http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="my-unique-persistence-unit-name">
        <properties>
            // properties...
        </properties>
    </persistence-unit>
</persistence>
```

For working examples of JPA entities, see the bmt, cmt, and hibernate5 quickstarts that ship with JBoss EAP 7.
5.10. JPA PERSISTENCE PROPERTY CHANGES

A new persistence property, `jboss.as.jpa.deferdetach`, was added to provide compatibility with the persistence behavior in previous releases of JBoss EAP.

The `jboss.as.jpa.deferdetach` property controls whether the transaction-scoped persistence context used in a non-JTA transaction thread detaches loaded entities after each `EntityManager` invocation or whether it waits until the persistence context is closed, for example, when the session bean invocation ends. The property value defaults to `false`, meaning entities are detached or cleared after each `EntityManager` invocation. This is the correct default behavior as defined in the JPA specification. If the property value is set to `true`, the entities are not detached until the persistence context is closed.

In JBoss EAP 5, persistence behaved as if the `jboss.as.jpa.deferdetach` property was set to `true`. To get this same behavior when migrating your application from JBoss EAP 5 to JBoss EAP 7, you must set the `jboss.as.jpa.deferdetach` property value to `true` in your `persistence.xml` as shown in the following example.

```xml
<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="1.0">
  <persistence-unit name="EAP5_COMPAT_PU">
    <jta-data-source>java:jboss/datasources/ExampleDS</jta-data-source>
    <properties>
      <property name="jboss.as.jpa.deferdetach" value="true"/>
    </properties>
  </persistence-unit>
</persistence>
```

In JBoss EAP 6, persistence behaved as if the `jboss.as.jpa.deferdetach` property was set to `false`. This is the same behavior as seen in JBoss EAP 7, so no changes are necessary when you migrate your application.

5.11. MIGRATE EJB CLIENT CODE

The default remote connector and port has changed in JBoss EAP 7. For details about this change, see Update the Remote URL Connector and Port.

If you used the `migrate` operation to migrate your server configuration, the old settings are preserved and you do not need to make the changes detailed below. However, if you run with the new JBoss EAP 7 default configuration, you must make the following changes.

5.11.1. Update the Default Remote Connection Port

Change the remote connection port value from 4447 to 8080 in the `jboss-ejb-client.properties` file.

The following are examples of a `jboss-ejb-client.properties` file in the previous and the current release.

**Example: JBoss EAP 6 jboss-ejb-client.properties file**

```properties
remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED=false
remote.connections=default
```
remote.connection.default.host=localhost
remote.connection.default.port=4447
remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOAUTHENTICATE=NONYMOUS=false

Example: JBoss EAP 7 jboss-ejb-client.properties file

remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED=false
remote.connections=default
remote.connection.default.host=localhost
remote.connection.default.port=8080
remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOAUTHENTICATE=NONYMOUS=false

Example: JBoss EAP 7 jboss-ejb-client.properties file

remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED=false
remote.connections=default
remote.connection.default.host=localhost
remote.connection.default.port=8080
remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOAUTHENTICATE=NONYMOUS=false

5.11.2. Update the Default Connector

If you are running with the new JBoss EAP 7 configuration, the default connector has changed from remote to http-remoting. This change impacts clients that use libraries from one release of JBoss EAP and connect to a server in a different release.

- If a client application uses the EJB client library from JBoss EAP 6 and wants to connect to JBoss EAP 7 server, the server must be configured to expose a remote connector on a port other than 8080. The client must then connect using that newly configured connector.

- A client application that uses the EJB client library from JBoss EAP 7 and wants to connect to JBoss EAP 6 server must be aware that the server instance does not use the http-remoting connector and instead uses a remote connector. This is achieved by defining a new client-side connection property.

remote.connection.default.protocol=remote

5.11.3. Migrate Remote Naming Client Code

If you are running with the new default JBoss EAP 7 configuration, you must modify your client code to use the new default remote port and connector.

The following is an example of how remote naming properties were specified in the client code in JBoss EAP 6.

```
java.naming.factory.initial=org.jboss.naming.remote.client.InitialContextFactory
java.naming.provider.url=remote://localhost:4447
```

The following is an example of how to specify the remote naming properties in the client code in JBoss EAP 7.

```
java.naming.factory.initial=org.jboss.naming.remote.client.InitialContextFactory
java.naming.provider.url=http-remoting://localhost:8080
```

5.12. MIGRATE DEPLOYMENT PLAN CONFIGURATIONS
The Java EE Application Deployment specification (JSR-88) was intended to define a standard contract to enable tools from multiple providers to configure and deploy applications on any Java EE platform product. The contract required Java EE Product Providers to implement the DeploymentManager and other javax.enterprise.deploy.spi interfaces to be accessed by the Tool Providers. In case of JBoss EAP 6, a deployment plan is identified by an XML descriptor named deployment-plan.xml that is bundled in a ZIP or JAR archive.

This specification saw very little adoption because most application server products provide their own more "feature rich" deployment solutions. For this reason, JSR-88 support was dropped from Java EE 7, and in turn, dropped from JBoss EAP 7.

If you used JSR-88 to deploy your application, you must now use another method to deploy the application. The JBoss EAP management CLI deploy command provides a standard way to deploy archives to standalone servers or to server groups in a managed domain. For more information about the management CLI, see the Management CLI Guide.

5.13. MIGRATE CUSTOM APPLICATION VALVES

You must manually migrate custom valves or any valves that are defined in the jboss-web.xml XML file. This includes valves created by extending the org.apache.catalina.valves.ValveBase class and configured in the <valve> element of the jboss-web.xml descriptor file.

IMPORTANT

Custom valves and valves that are defined in the jboss-web.xml file must be rewritten or replaced by the corresponding Undertow built-in handler. For information about mapping valves to Undertow handlers, see Migrate JBoss Web Valves.

Authentication valves must be replaced manually using Undertow built-in authentication mechanisms.

Migrate Valves Configured in Deployments

In JBoss EAP 6, you could define custom valves at the application level by configuring them in the jboss-web.xml web application descriptor file. In JBoss EAP 7, it is possible to do this with Undertow handlers as well.

The following is an example of a valve configured in the jboss-web.xml file in JBoss EAP 6.

```xml
<jboss-web>
  <valve>
    <class-name>org.jboss.examples.MyValve</class-name>
    <param>
      <param-name>myParam</param-name>
      <param-value>foobar</param-value>
    </param>
  </valve>
</jboss-web>
```

For more information about how to create and configure custom handlers in JBoss EAP, see Creating Custom Handlers in the JBoss EAP Development Guide.

Migrate Custom Authenticator Valves

For information about how to migrate authenticator valves, see Security Application Changes.
5.14. SECURITY APPLICATION CHANGES

The replacement of JBoss Web with Undertow requires changes to security configuration in JBoss EAP 7.

5.14.1. Migrate Authenticator Valves

Authentication valves must be replaced manually using Undertow built-in authentication mechanisms. See the following sections for information about how to migrate authenticator valves.

5.14.2. PicketLink Changes

For information about the changes required for SSO with SAML v2 configuration, see Changes from Previous Versions of JBoss EAP in How to Set Up SSO with SAML v2 for JBoss EAP.

5.14.3. Other Security Application Changes

For information about the differences in SSO configuration with Kerberos, see Differences from Configuring Previous Versions JBoss EAP in How to Set Up SSO with Kerberos for JBoss EAP.

5.15. JBOSS LOGGING CHANGES

If your application uses JBoss Logging, be aware that the annotations in the org.jboss.logging package are now deprecated in JBoss EAP 7. They have been moved to the org.jboss.logging.annotations package, so you must update your source code to import the new package.

The annotations have also moved to a separate Maven groupId:artifactId:version (GAV) ID so you need to add a new project dependency for org.jboss.logging:jboss-logging-annotations in your project pom.xml file.

**NOTE**

Only the logging annotations have moved. The org.jboss.logging.BasicLogger and org.jboss.logging.Logger still exist in the org.jboss.logging package.

The following table lists the deprecated annotation classes and corresponding replacements.

**Table 5.1. Deprecated Logging Annotation Replacements**

<table>
<thead>
<tr>
<th>Deprecated Class</th>
<th>Replacement Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.jboss.logging.Cause</td>
<td>org.jboss.logging.annotations.Cause</td>
</tr>
<tr>
<td>org.jboss.logging.Field</td>
<td>org.jboss.logging.annotations.Field</td>
</tr>
<tr>
<td>org.jboss.logging.FormatWith</td>
<td>org.jboss.logging.annotations.FormatWith</td>
</tr>
<tr>
<td>org.jboss.logging.LoggingClass</td>
<td>org.jboss.logging.annotations.LoggingClass</td>
</tr>
<tr>
<td>Deprecated Class</td>
<td>Replacement Class</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>org.jboss.logging.MessageBundle</td>
<td>org.jboss.logging.annotations.MessageBundle</td>
</tr>
<tr>
<td>org.jboss.logging.Param</td>
<td>org.jboss.logging.annotations.Param</td>
</tr>
<tr>
<td>org.jboss.logging.Property</td>
<td>org.jboss.logging.annotations.Property</td>
</tr>
</tbody>
</table>

### 5.16. JAVASERVER FACES (JSF) CODE CHANGES

**Dropped Support for JSF 1.2**

JBoss EAP 6 allowed you to continue to use JSF 1.2 with your application deployment by creating a `jboss-deployment-structure.xml` file.

JBoss EAP 7 includes JSF 2.2 and no longer supports the JSF 1.2 API. If your application uses JSF 1.2, you must rewrite it to use JSF 2.2.

**Compatibility Issue Between JSF 2.1 and JSF 2.2**

The JSF 2.1 and JSF 2.2 APIs are not fully compatible. The `FACELET_CONTEXT_KEY` constant value changed from `com.sun.faces.facelets.FACELET_CONTEXT` to `javax.faces.FACELET_CONTEXT` between the releases. This value is inlined by the compiler and code compiled against one release will not work against the other.

Applications that contain code similar to the following example, and are compiled with the JSF 2.1 API but are run in JBoss EAP 7, which uses the JSF 2.2 API, result in a `NullPointerException`. To fix the problem, you must recompile the application against the JSF 2.2 API.

```java
Object obj = FacesContext.getCurrentInstance().getAttributes().get(FaceletContext.FACELET_CONTEXT_KEY);
```

See [Prevent FaceletContext.FACELET_CONTEXT_KEY constant to be inlined by compiler](#) for more information.

### 5.17. MODULE CLASS LOADING CHANGES

In JBoss EAP 7, the class loading behavior has changed in cases where multiple modules contain the same classes or packages.

Assume there are two modules, `MODULE_A` and `MODULE_B`, that depend upon each other and contain some of the same packages. In JBoss EAP 6, the classes or packages that were loaded from the dependencies took precedence over those specified in the `resource-root` of the `module.xml` file. This meant `MODULE_A` saw the packages for `MODULE_B` and `MODULE_B` saw the packages for `MODULE_A`. This behavior was confusing and could cause conflicts. This behavior has changed in JBoss EAP 7. Now the classes or packages specified by the `resource-root` in the `module.xml` file take
precedence over those specified by the dependency. This means MODULE_A sees the packages for MODULE_A and MODULE_B sees the packages for MODULE_B. This prevents conflicts and provides a more appropriate behavior.

If you have defined custom modules that include resource-root libraries or packages that contain classes that are duplicated in their module dependencies, you might see ClassCastException, LinkageError, class loading errors, or other changes in behavior when you migrate to JBoss EAP 7. To resolve these issues, you must configure your module.xml file to ensure only one version of a class is used. This can be accomplished by using either of the following approaches.

- You can avoid specifying a resource-root that duplicates classes in the module dependency.
- You can use the include and exclude sub-elements of the imports and exports elements to control class loading in the module.xml file. The following is an export element that excludes classes is in the specified package.

```xml
<exports>
  <exclude path="com/mycompany/duplicateclassespath"/>
</exports>
```

If you prefer to preserve your existing behavior, you must filter the dependency packages from the dependent resource-root in the module.xml file using the filter element. This allows you to retain the existing behavior without the odd looping that you would see under JBoss EAP 6. The following is an example of a root-resource that filters classes in a specified package.

```xml
<resource-root path="mycompany.jar">
  <filter>
    <exclude path="com/mycompany/duplicateclassespath"/>
  </filter>
</resource-root>
```

For more information about modules and class loading, see Class Loading and Modules in the JBoss EAP Development Guide.

### 5.18. APPLICATION CLUSTERING CHANGES

#### 5.18.1. Overview of New Clustering Features

The following list describes some of the new clustering features to be aware of when migrating your application from JBoss EAP 6 to JBoss EAP 7.

- JBoss EAP 7 introduces a new public API for building singleton services that significantly simplifies the process. For information on singleton services, see HA Singleton Service in the JBoss EAP Development Guide.

- A singleton deployment can be configured to deploy and start on only a single node in the cluster at a time. For more information, see HA Singleton Deployments in the JBoss EAP Development Guide.

- You can now define clustered singleton MDBs. For more information, see Clustered Singleton MDBs in Developing EJB Applications for JBoss EAP.
JBoss EAP 7 includes the Undertow mod_cluster implementation. This offers a pure Java load balancing solution that does not require an httpd web server. For more information, see Configuring JBoss EAP as a Front-end Load Balancer in the JBoss EAP Configuration Guide.

The remainder of this section describes how clustering changes might impact the migration of your applications to JBoss EAP 7.

### 5.18.2. Web Session Clustering Changes

JBoss EAP 7 introduces a new web session clustering implementation. It replaces the previous implementation, which was tightly coupled to the legacy JBoss Web subsystem source code.

The new web session clustering implementation impacts how the application is configured in the jboss-web.xml JBoss proprietary web application XML descriptor file. The following are the only clustering configuration elements that remain in this file.

```xml
<jboss-web>
  ...
  <max-active-sessions>...</max-active-sessions>
  ...
  <replication-config>
    <replication-granularity>...</replication-granularity>
    <cache-name>...</cache-name>
  </replication-config>
  ...
</jboss-web>
```

The following table describes how to achieve similar behavior for elements in the jboss-web.xml file that are now obsolete.

<table>
<thead>
<tr>
<th>Configuration Element</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;max-active-sessions/&gt;</code></td>
<td>Previously, the session creation would fail if it caused the number of active sessions to exceed the value specified by <code>&lt;max-active-sessions/&gt;</code>. In the new implementation, <code>&lt;max-active-sessions/&gt;</code> is used to enable session passivation. If session creation will cause the number of active sessions to exceed the <code>&lt;max-active-sessions/&gt;</code>, then the oldest session known to the session manager will passivate to make room for the new session.</td>
</tr>
<tr>
<td><code>&lt;passivation-config/&gt;</code></td>
<td>This configuration element and its sub-elements are no longer used in JBoss EAP 7.</td>
</tr>
<tr>
<td><code>&lt;use-session-passivation/&gt;</code></td>
<td>Previously, passivation was enabled using this attribute. In the new implementation, passivation is enabled by specifying a non-negative value for <code>&lt;max-active-sessions/&gt;</code>.</td>
</tr>
<tr>
<td>Configuration Element</td>
<td>Description of Change</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td><code>&lt;passivation-min-idle-time/&gt;</code></td>
<td>Previously, sessions needed to be active for a minimum amount of time before becoming a candidate for passivation. This could cause session creation to fail, even when passivation was enabled. The new implementation does not support this logic and thus avoids this Denial of Service (DoS) vulnerability.</td>
</tr>
<tr>
<td><code>&lt;passivation-max-idle-time/&gt;</code></td>
<td>Previously, a session would be passivated after it was idle for a specific amount of time. The new implementation only supports lazy passivation. It does not support eager passivation. Sessions are only passivated when necessary to comply with <code>&lt;max-active-sessions/&gt;</code>.</td>
</tr>
<tr>
<td><code>&lt;replication-config/&gt;</code></td>
<td>The new implementation deprecates a number of sub-elements.</td>
</tr>
<tr>
<td><code>&lt;replication-trigger/&gt;</code></td>
<td>Previously, this element was used to determine when session replication was triggered. The new implementation replaces this configuration option with a single, robust strategy. For more information, see Immutable Session Attributes in the JBoss EAP Development Guide.</td>
</tr>
<tr>
<td><code>&lt;use-jk/&gt;</code></td>
<td>Previously, the <code>instance-id</code> of the node handling a given request was appended to the <code>jsessionid</code>, for use by load balancers such as <code>mod_jk</code>, <code>mod_proxy_balancer</code>, <code>mod_cluster</code>, depending on the value specified for <code>&lt;use-jk/&gt;</code>. In the new implementation, the <code>instance-id</code>, if defined, is always appended to the <code>jsessionid</code>.</td>
</tr>
<tr>
<td><code>&lt;max-unreplicated-interval/&gt;</code></td>
<td>Previously, this configuration option was intended as an optimization to prevent the replication of a session’s timestamp if no session attribute was changed. While this sounds nice, in practice it does not prevent any RPCs, since session access requires cache transaction RPCs regardless of whether any session attributes changed. In the new implementation, the timestamp of a session is replicated on every request. This prevents stale session metadata following a failover.</td>
</tr>
<tr>
<td><code>&lt;snapshot-mode/&gt;</code></td>
<td>Previously, one could configure <code>&lt;snapshot-mode/&gt;</code> as <code>INSTANT</code> or <code>INTERVAL</code>. Infinispan’s asynchronous replication makes this configuration option obsolete.</td>
</tr>
<tr>
<td><code>&lt;snapshot-interval/&gt;</code></td>
<td>This was only relevant for <code>&lt;snapshot-mode&gt;INTERVAL&lt;/snapshot-mode&gt;</code>. Since <code>&lt;snapshot-mode/&gt;</code> is obsolete, this option is now obsolete as well.</td>
</tr>
</tbody>
</table>
Previously, the value specified by this attribute defined a policy for triggering session events.
In the new implementation, this behavior is specification-driven and not configurable.

<table>
<thead>
<tr>
<th>Configuration Element</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;session-notification-policy/&gt;</td>
<td>Previously, the value specified by this attribute defined a policy for triggering session events. In the new implementation, this behavior is specification-driven and not configurable.</td>
</tr>
</tbody>
</table>

This new implementation also supports write-through cache stores as well as passivation-only cache stores. Typically, a write-through cache store is used in conjunction with an invalidation cache. The web session clustering implementation in JBoss EAP 6 did not operate correctly when used with an invalidation cache.

### 5.18.3. Stateful Session EJB Clustering Changes

In JBoss EAP 6, you were required to enable the clustering behavior for stateful session beans (SFSBs) in one of the following ways.

- You could add the `org.jboss.ejb3.annotation.Clustered` annotation in the session bean.

  ```java
  @Stateful
  @Clustered
  public class MyBean implements MySessionInt {
    public void myMethod() {
      //
    }
  }
  ```

- You could add the `<clustered>` element to the `jboss-ejb3.xml` file.

  ```xml
  <c:clustering>
    <ejb-name>DDBasedClusteredSFSB</ejb-name>
    <c:clustered>true</c:clustered>
  </c:clustering>
  ```

JBoss EAP 7 no longer requires you to enable the clustering behavior. By default, if the server is started using an HA profile, the state of SFSBs will be replicated automatically.

You can disable this default behavior in one of the following ways.

- You can disable the default behavior for a single stateful session bean by using `@Stateful(passivationCapable=false)`, which is new to the EJB 3.2 specification.

- You can disable this behavior globally in the configuration of the `ejb3` subsystem in the server configuration.

**NOTE**

If the `@Clustered` annotation is not removed from the application, it is simply ignored and does not affect the deployment of the application.
5.18.4. Clustering Services Changes

In JBoss EAP 6, the APIs for clustering services were in private modules and were not supported.

JBoss EAP 7 introduces a public clustering services API for use by applications. The new services are designed to be lightweight, easily injectable, and require no external dependencies.

- The new `org.wildfly.clustering.group.Group` interface provides access to the current cluster status and allows listening for cluster membership changes.

- The new `org.wildfly.clustering.dispatcher.CommandDispatcher` interface allows running code in the cluster, on all or a selected subset of nodes.

These services replace similar APIs that were available in previous releases, namely HAPartition from JBoss EAP 5 and GroupCommunicationService, GroupMembershipNotifier, and GroupRpcDispatcher in JBoss EAP 6.

For more information, see Public API for Clustering Services in the JBoss EAP Development Guide.

5.18.5. Migrate Clustering HA Singleton

In JBoss EAP 6, there was no public API available for the cluster-wide HA singleton service. If you used the private `org.jboss.as.clustering.singleton.*` classes, you must change your code to use the new public `org.wildfly.clustering.singleton.*` packages when you migrate your application to JBoss EAP 7.

For more information about how to implement an HA singleton, see HA Singleton Service in the Development Guide for JBoss EAP.
CHAPTER 6. MISCELLANEOUS CHANGES

6.1. CHANGES TO DELIVERY OF JBOSS EAP NATIVES AND APACHE HTTP SERVER

JBoss EAP 7 natives are delivered differently in this release than in the past. Some now ship with the new Red Hat JBoss Core Services product, which is a set of supplementary software that is common to many of the Red Hat JBoss middleware products. The new product allows for faster distribution of updates and a more consistent update experience. The JBoss Core Services product is available for download in a different location on the Red Hat Customer Portal.

- The following table lists the differences in the delivery methods between the releases.

<table>
<thead>
<tr>
<th>Package</th>
<th>JBoss EAP 6</th>
<th>JBoss EAP 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIO Natives for Messaging</td>
<td>Delivered with the product in a separate &quot;Native Utilities&quot; download</td>
<td>Included within the JBoss EAP distribution. No additional download is required.</td>
</tr>
<tr>
<td>Apache HTTP Server</td>
<td>Delivered with the product in a separate &quot;Apache HTTP Server&quot; download</td>
<td>Delivered with the new JBoss Core Services product</td>
</tr>
<tr>
<td>mod_cluster, mod_jk, isapi, and nsapi connectors</td>
<td>Delivered with the product in a separate &quot;Webserver Connector Natives&quot; download</td>
<td>Delivered with the new JBoss Core Services product</td>
</tr>
<tr>
<td>JSVC</td>
<td>Delivered with the product in a separate &quot;Native Utilities&quot; download</td>
<td>Delivered with the new JBoss Core Services product</td>
</tr>
<tr>
<td>OpenSSL</td>
<td>Delivered with the product in a separate &quot;Native Utilities&quot; download</td>
<td>This was dropped in JBoss EAP 7</td>
</tr>
<tr>
<td>tcnatives</td>
<td>Delivered with the product in a separate &quot;Native Components&quot; download</td>
<td>This was dropped in JBoss EAP 7</td>
</tr>
</tbody>
</table>

- You should also be aware of the following changes:
  - Support was dropped for mod_cluster and mod_jk connectors used with Apache HTTP Server from Red Hat Enterprise Linux RPM channels. If you run Apache HTTP Server from Red Hat Enterprise Linux RPM channels and need to configure load balancing for JBoss EAP 7 servers, you can do one of the following:
    - Use the Apache HTTP Server provided by JBoss Core Services.
    - You can configure JBoss EAP 7 to act as a front-end load balancer. For more information, see Configuring JBoss EAP as a Front-end Load Balancer in the JBoss EAP Configuration Guide.
    - You can deploy Apache HTTP Server on a machine that is supported and certified and then run the load balancer on that machine. For the list of supported configurations,
Support was dropped for mod_cluster and mod_jk connectors used with Apache HTTP Server from the HP-UX Web Server Suites. If you run Apache HTTP Server from HP-UX Web Server Suites and need to configure load balancing for JBoss EAP 7 servers, you can do one of the following:

- You can configure JBoss EAP 7 to act as a front-end load balancer. For more information, see Configuring JBoss EAP as a Front-end Load Balancer in the JBoss EAP Configuration Guide.

- You can deploy Apache HTTP Server on a machine that is supported and certified and then run the load balancer on that machine. For the list of supported configurations, see Overview of HTTP Connectors in the JBoss EAP Configuration Guide.

You can find more information about JBoss Core Services in the Apache HTTP Server Installation Guide.

### 6.2. CHANGES TO DEPLOYMENTS ON AMAZON EC2

A number of changes have been made to the Amazon Machine Images (AMI) in JBoss EAP 7. This section briefly summarizes some of those changes.

- The way you launch non-clustered and clustered JBoss EAP instances and domains in Amazon EC2 has changed significantly.

- JBoss EAP 6 used the User Data: field for JBoss EAP configuration. The AMI scripts that parsed the configuration in the User Data: field and started the servers automatically on instance startup have been removed from JBoss EAP 7.

- Red Hat JBoss Operations Network agent was installed in the previous release of JBoss EAP. In JBoss EAP 7, you must install it separately.

For details on deploying JBoss EAP 7 on Amazon EC2, see Deploying Red Hat JBoss Enterprise Application Platform on Amazon EC2.

### 6.3. CHANGES TO JBOSS EAP SCRIPTS

The add-user script behavior has changed in JBoss EAP 7 due to a change in password policy. JBoss EAP 6 had a strict password policy. As a result, the add-user script rejected weak passwords that did not satisfy the minimum requirements. In JBoss EAP 7, weak passwords are accepted and a warning is issued. For more information, see Setting Add-User Utility Password Restrictions in the JBoss EAP Configuration Guide.

### 6.4. REMOVAL OF OSGI SUPPORT

When JBoss EAP 6.0 GA was first released, JBoss OSGi, an implementation of the OSGi specification, was included as a Technology Preview feature. With the release of JBoss EAP 6.1.0, JBoss OSGi was demoted from Technology Preview to Unsupported.

In JBoss EAP 6.1.0, the configadmin and osgi extension modules and subsystem configuration for a standalone server were moved to a separate file. EAP_HOME/standalone/configuration/standalone osgi.xml configuration file.
should not migrate this unsupported configuration file, the removal of JBoss OSGi support should not impact the migration of a standalone server configuration. If you modified any of the other standalone configuration files to configure osgi or configadmin, those configurations must be removed.

For a managed domain, the osgi extension and subsystem configuration were removed from the EAP_HOME/domain/configuration/domain.xml file in the JBoss EAP 6.1.0 release. However, the configadmin module extension and subsystem configuration remain in the EAP_HOME/domain/configuration/domain.xml file. This configuration is no longer supported in JBoss EAP 7 and must be removed.
CHAPTER 7. MIGRATING FROM OLDER RELEASES OF JBOSS EAP

7.1. MIGRATING FROM JBOSS EAP 5 TO JBOSS EAP 7

This guide focuses on the changes that are required to successfully run and deploy JBoss EAP 6 applications on JBoss EAP 7. If you plan to migrate your applications directly from JBoss EAP 5 to JBoss EAP 7, there are a number of resources available to help you plan and execute your migration. We suggest you take the following approach.

1. See Summary of Changes Made to Each Release in this guide for a quick, high-level overview of the changes made to each release of JBoss EAP.

2. Read through the JBoss EAP 6 Migration Guide and this guide to become familiar with the contents of each one.

3. Use the JBoss EAP 5 Component Upgrade Reference as a quick reference to migration information about specific components and features.

4. The rule-based Red Hat Application Migration Toolkit continues to add rules to help you migrate directly from JBoss EAP 5 to JBoss EAP 7. You can use these tools to analyze your application and to generate detailed reports about the changes needed to migrate to JBoss EAP 7. For more information, see Use Red Hat Application Migration Toolkit to Analyze Applications for Migration.

5. The Customer Portal Knowledgebase currently contains articles and solutions to help with migration from JBoss EAP 5 to JBoss EAP 6. There are plans in place to add additional content for migration from JBoss EAP 5 to JBoss EAP 7 over time.

7.2. SUMMARY OF CHANGES MADE TO EACH RELEASE

Before you plan your migration, you should be aware of the changes that were made to JBoss EAP 6 and JBoss EAP 7.

The JBoss EAP 6 Migration Guide covers changes that were made between JBoss EAP 5 and JBoss EAP 6. The following is a condensed list of the most significant changes made in JBoss EAP 6.

- Implemented a new architecture built on the Modular Service Container
- Was a certified implementation of the Java Enterprise Edition 6 specification
- Introduced domain management, new deployment configuration, and a new file directory structure and scripts
- Standardized on new portable JNDI namespaces

See Review What’s New and Different in JBoss EAP 6 in the JBoss EAP 6 Migration Guide for a detailed list of changes made in that release.

JBoss EAP 7 is built on the same modular structure as JBoss EAP 6 and includes the same domain management, deployment configuration, file directory structure, and scripts. It also still uses the same standardized JNDI namespaces. However, JBoss EAP 7 introduces the following changes.

- Adds support for the Java Enterprise Edition 7 specification
● Replaces the web server with Undertow

● Replaces the JacORB IIOP implementation with a downstream branch of the OpenJDK ORB

● Includes Apache ActiveMQ Artemis as the new messaging provider

● Removes the cmp, jaxr, and threads subsystems

● Removes support for EJB entity beans

For a more complete list of changes, see Review What’s New in JBoss EAP 7

7.3. REVIEW THE CONTENT IN THE MIGRATION GUIDES

Review the entire contents of the Migration Guide for each release to become aware of the features that were added or deprecated, and to understand the server configuration and the application changes required to run existing applications for that release.

Because the underlying architecture was not changed between JBoss EAP 6 and JBoss EAP 7, many of the changes documented in the JBoss EAP 6 Migration Guide still apply. For example, changes documented under Changes Required by Most Applications are related to the underlying architectural changes made in JBoss EAP 6, which still apply to this release. The change to the new modular class loading system is significant and impacts the packaging and dependencies of almost every JBoss EAP 5 application. Many of the changes listed under Changes Dependent on Your Application Architecture and Components are also still valid. However, because JBoss EAP 7 replaced the web server, ORB, and messaging provider, removed the cmp, threads, and jaxr subsystems, and removed support for EJB entity beans, you must consult this guide for any changes related to those component areas. Pay particular attention to the Server Configuration Changes and Application Migration Changes detailed in this guide before you begin.

7.4. JBOSS EAP 5 COMPONENT UPGRADE REFERENCE

Use the following table to find information about how to migrate a particular feature or component from JBoss EAP 5 to JBoss EAP 7.

<table>
<thead>
<tr>
<th>JBoss EAP 5 Feature or Component</th>
<th>Summary of Changes and Where to Find Migration Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Packaging and Class Loading</td>
<td>In JBoss EAP 6, the previous hierarchical class loading structure was replaced with a modular architecture based on JBoss Modules. Application packaging also changed due to the new modular class loading structure. This architecture is still used in JBoss EAP 7. For information about the new modular architecture, see the following chapter in the JBoss EAP 7 Development Guide.</td>
</tr>
<tr>
<td></td>
<td>● Class Loading and Modules</td>
</tr>
<tr>
<td></td>
<td>For information about how to update and repackage applications for the new modular architecture, see the following section in the JBoss EAP 6 Migration Guide.</td>
</tr>
<tr>
<td></td>
<td>● Class Loading Changes</td>
</tr>
<tr>
<td>JBoss EAP 5 Feature or Component</td>
<td>Summary of Changes and Where to Find Migration Information</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Application Configuration Files</strong></td>
<td>Due to the changes in JBoss EAP 6 to use modular class loading, you might need to create or modify one or more application configuration files to add dependencies or to prevent automatic dependencies from loading. This has not changed in JBoss EAP 7. For details, see the following section in the JBoss EAP 6 Migration Guide.</td>
</tr>
<tr>
<td><strong>Caching and Infinispan</strong></td>
<td>JBoss Cache was replaced by Infinispan for internal use by the server only in JBoss EAP 6. See the following sections in the JBoss EAP 6 Migration Guide for information about how to replace JBoss Cache in application code.</td>
</tr>
<tr>
<td><strong>Data Sources and Resource Adapters</strong></td>
<td>JBoss EAP 6 consolidated configuration of data sources and resource adapters into mainly one file and this is still true in JBoss EAP 7. See the following section in the JBoss EAP 6 Migration Guide for more information.</td>
</tr>
<tr>
<td><strong>Directory Structure, Scripts, and Deployment Configuration</strong></td>
<td>In JBoss EAP 6, the directory structure, scripts, and deployment configuration changed. These changes are still valid in JBoss EAP 7. See the following section of the JBoss EAP 6 Migration Guide for more information.</td>
</tr>
</tbody>
</table>

- **Configuration File Changes**
- **Cache Changes**
- **Infinispan Caching Strategy and Configuration Changes**
- **Datasource and Resource Adapter Configuration Changes**
- **Review The List of Deprecated and Unsupported Features**
- **Review What's New and Different in JBoss EAP 6**
<table>
<thead>
<tr>
<th>JBoss EAP 5 Feature or Component</th>
<th>Summary of Changes and Where to Find Migration Information</th>
</tr>
</thead>
</table>
| EJB                              | The Java EE 7 specification made EJB 2.x and earlier features optional, so it is strongly recommended that you rewrite your application code to use the EJB 3.x specification and JPA. For information about deprecated features and changes required to run EJB 2.x, see the following section in the JBoss EAP 6 Migration Guide.  
  - EJB 2.x and Earlier Changes  
    In JBoss EAP 6, stateful EJB cache and stateless session bean pool size is configured in the `ejb3` subsystem of the server configuration file. The `jboss-ejb3.xml` deployment descriptor replaces the `jboss.xml` deployment descriptor file. For more information about these changes, see the following section in the JBoss EAP 6 Migration Guide.  
    - EJB Changes  
      The default remote connector and port has changed in JBoss EAP 7. For more information about this and server configuration changes, see the following sections in this guide.  
      - EJB Server Configuration Changes  
      - Migrate EJB Client Code  
      EJB entity beans are not supported in JBoss EAP 7. For information about how to migrate entity beans to JPA, see the following section in this guide.  
      - Migrate Entity Beans to JPA |
| Hibernate and JPA                | In JBoss EAP 6, Hibernate was updated from version 3 to version 4. This version of JBoss EAP also implemented the JPA 2.0 specification and changes were made to JPA persistence properties. For information about how to modify your application for these changes, see the following section in the JBoss EAP 6 Migration Guide.  
  - Hibernate and JPA Changes  
    JBoss EAP 7 implements JPA 2.1 and includes Hibernate 5. It also updates Hibernate Search from version 4.6.x to version 5.5.x. Other changes include removal of support for EJB entity beans and additional updates to JPA persistence properties. For information about how these changes impact your applications, see the following sections in this guide.  
    - Hibernate and JPA Migration Changes  
    - Hibernate Search Changes  
    - Migrate Entity Beans to JPA  
    - JPA Persistence Property Changes |
<table>
<thead>
<tr>
<th>JBoss EAP 5 Feature or Component</th>
<th>Summary of Changes and Where to Find Migration Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JAX-RS and RESTEasy</strong></td>
<td>JBoss EAP 6 bundled RESTEasy 2, which automatically configured RESTEasy and required changes in application configuration. See the following section in the JBoss EAP 6 Migration Guide for information.</td>
</tr>
<tr>
<td></td>
<td>- JAX-RS and RESTEasy Changes</td>
</tr>
<tr>
<td></td>
<td>JBoss EAP 7 includes RESTEasy 3 and many classes have been deprecated. The version of Jackson changed from version 1.9.9 to version 2.6.3 or greater. For details about these changes, see the following section in this guide.</td>
</tr>
<tr>
<td></td>
<td>- JAX-RS and RESTEasy Application Changes</td>
</tr>
<tr>
<td><strong>JBoss AOP</strong></td>
<td>JBoss AOP (Aspect Oriented Programming) was removed in JBoss EAP 6. For information about how to refactor applications that use JBoss AOP, see the following section in the JBoss EAP 6 Migration Guide.</td>
</tr>
<tr>
<td></td>
<td>- JBoss AOP Changes</td>
</tr>
<tr>
<td><strong>JGroups and Clustering</strong></td>
<td>The way you enable clustering and specify bind addresses changed in JBoss EAP 6. See the following section in the JBoss EAP 6 Migration Guide for more information.</td>
</tr>
<tr>
<td></td>
<td>- Clustering Changes</td>
</tr>
<tr>
<td></td>
<td>In JBoss EAP 7, JGroups now defaults to using a private network interface instead of a public network interface and also introduces <code>&lt;channel&gt;</code> elements to the <code>jgroups</code> subsystem. JBoss EAP 7 also includes the Undertow <code>mod_cluster</code> implementation, introduces a new API for building singleton services, and other new clustering features. These changes are documented in the following sections of this guide.</td>
</tr>
<tr>
<td></td>
<td>- JGroups Server Configuration Changes</td>
</tr>
<tr>
<td></td>
<td>- Application Clustering Changes</td>
</tr>
<tr>
<td><strong>JNDI</strong></td>
<td>JBoss EAP 6 implemented a new standardized global JNDI namespace and a series of related namespaces that map to the various scopes of a Java EE application. See the following section of the JBoss EAP 6 Migration Guide for information about application changes needed to use the new JNDI namespace rules.</td>
</tr>
<tr>
<td></td>
<td>- JNDI Changes</td>
</tr>
<tr>
<td>JBoss EAP 5 Feature or Component</td>
<td>Summary of Changes and Where to Find Migration Information</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>JSF</td>
<td>JBoss EAP 6 included JSF 2.0 and allowed you to configure your application to use an older version. This is no longer possible in JBoss EAP 7, which now includes JSF 2.2. See the following section in this guide for more information.</td>
</tr>
<tr>
<td></td>
<td>- JavaServer Faces (JSF) Code Changes</td>
</tr>
<tr>
<td>Logging</td>
<td>JBoss EAP 6 introduced a new JBoss Logging framework that is still used in JBoss EAP 7. Applications that use third-party logging frameworks might be impacted by the modular class loading changes. Review the following section in the JBoss EAP 6 Migration Guide for information about these changes.</td>
</tr>
<tr>
<td></td>
<td>- Logging Changes</td>
</tr>
<tr>
<td></td>
<td>In JBoss EAP 7, annotations in the org.jboss.logging package are now deprecated, which impacts source code and Maven GAVs (groupId:artifactId:version). The prefixes for all log messages were also changed. For more information about these changes, see the following sections in this guide.</td>
</tr>
<tr>
<td></td>
<td>- JBoss Logging Changes</td>
</tr>
<tr>
<td></td>
<td>- Logging Message Prefix Changes</td>
</tr>
<tr>
<td>Messaging and JMS</td>
<td>In JBoss EAP 6, HornetQ replaced JBoss Messaging as the default JMS implementation. Then in JBoss EAP 7, ActiveMQ Artemis replaced HornetQ as the built-in messaging provider.</td>
</tr>
<tr>
<td></td>
<td>The best approach to migrating your messaging configuration is to start with the JBoss EAP 7 default server configuration and use the following guide to apply your current messaging configuration changes.</td>
</tr>
<tr>
<td></td>
<td>- Configuring Messaging for JBoss EAP 7</td>
</tr>
<tr>
<td></td>
<td>If you want to understand the changes required to move from JBoss Messaging to HornetQ, review the following section of the JBoss EAP 6 Migration Guide.</td>
</tr>
<tr>
<td></td>
<td>- HornetQ Changes</td>
</tr>
<tr>
<td></td>
<td>Then review the following information about how to migrate the HornetQ configuration and related messaging data in this guide.</td>
</tr>
<tr>
<td></td>
<td>- Messaging Server Configuration Changes</td>
</tr>
<tr>
<td></td>
<td>- Messaging Application Changes</td>
</tr>
<tr>
<td>JBoss EAP 5 Feature or Component</td>
<td>Summary of Changes and Where to Find Migration Information</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>ORB</td>
<td>In JBoss EAP 6, JacORB configuration was moved from the \EAP_HOME/server/production/conf/jacorb.properties file to the server configuration file. JBoss EAP 7 then replaced the JacORB IIOP implementation with a downstream branch of the OpenJDK ORB. The best approach to migrating your ORB configuration is to start with the JBoss EAP 7 default server configuration and use the following section in the JBoss EAP 7 Configuration Guide to apply the your current ORB configuration changes.</td>
</tr>
<tr>
<td></td>
<td>• ORB Configuration</td>
</tr>
<tr>
<td>Remote Invocation</td>
<td>A new EJB client API was introduced in JBoss EAP 6 for remote invocations; however, if you preferred not to rewrite your application code to use the new API, you could modify your existing code to use the \ejb:BEAN_REFERENCE for remote access to EJBs. See the following section in the JBoss EAP 6 Migration Guide for more information.</td>
</tr>
<tr>
<td></td>
<td>• Remote Invocation Changes</td>
</tr>
<tr>
<td></td>
<td>In JBoss EAP 7, the default connector and default remote connection port changed. For more information, see the following sections in this guide.</td>
</tr>
<tr>
<td></td>
<td>• Update the Remote URL Connector and Port</td>
</tr>
<tr>
<td></td>
<td>• Update External Clients</td>
</tr>
<tr>
<td></td>
<td>• Migrate EJB Client Code</td>
</tr>
<tr>
<td>Seam 2.x</td>
<td>While official support for Seam 2.2 applications was dropped in JBoss EAP 6, it was still possible to configure dependencies for JSF 1.2 and Hibernate 3 to allow Seam 2.2 applications to run on that release. JBoss EAP 7, which now includes JSF 2.2 and Hibernate 5, does not support Seam 2.2 or Seam 2.3 due to end of life of Red Hat JBoss Web Framework Kit. It is recommended that you rewrite your Seam components using Weld CDI beans.</td>
</tr>
<tr>
<td>Security</td>
<td>Security updates in JBoss EAP 6 included changes to security domain names and changes to how to configure security for basic authentication. The LDAP security realm configuration was moved to the server configuration file. See the following sections in the JBoss EAP 6 Migration Guide for more information.</td>
</tr>
<tr>
<td></td>
<td>• Security Changes</td>
</tr>
<tr>
<td></td>
<td>• LDAP Security Realm Changes</td>
</tr>
<tr>
<td></td>
<td>Updates that impact security in JBoss EAP 7 include server configuration changes and application changes. Information can be found in the following sections of this guide.</td>
</tr>
<tr>
<td></td>
<td>• Security Server Configuration Changes</td>
</tr>
<tr>
<td></td>
<td>• Security Application Changes</td>
</tr>
<tr>
<td>JBoss EAP 5 Feature or Component</td>
<td>Summary of Changes and Where to Find Migration Information</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Spring Applications</td>
<td>Spring 4.2.x is the earliest stable Spring version supported by JBoss EAP 7. For information about Apache CXF Spring web services and Spring RESTEasy integration changes, see the following sections in this guide.</td>
</tr>
<tr>
<td></td>
<td>- Apache CXF Spring Web Services Changes</td>
</tr>
<tr>
<td></td>
<td>- Spring RESTEasy Integration Changes</td>
</tr>
<tr>
<td>Transactions</td>
<td>JBoss EAP 6 consolidated transaction configuration and moved it to the server configuration file. Other updates included changes to JTA node identifier settings and how to enable JTS. For details, see the following section in the JBoss EAP 6 Migration Guide.</td>
</tr>
<tr>
<td></td>
<td>- JTS and JTA Changes</td>
</tr>
<tr>
<td></td>
<td>Some Transaction Manager configuration attributes that were available in the transactions subsystem in JBoss EAP 6 have changed in JBoss EAP 7. For more information, see the following section in this guide.</td>
</tr>
<tr>
<td></td>
<td>- Migrate Transactions Subsystem Changes</td>
</tr>
<tr>
<td>Valves</td>
<td>Undertow replaced JBoss Web in JBoss EAP 7 and valves are no longer supported. See the following sections in this guide.</td>
</tr>
<tr>
<td></td>
<td>- Migrate Global Valves</td>
</tr>
<tr>
<td></td>
<td>- Migrate Custom Application Valves</td>
</tr>
<tr>
<td></td>
<td>- Migrate Authenticator Valves</td>
</tr>
<tr>
<td>Web Services</td>
<td>JBoss EAP 6 included JBossWS 4. For information about the changes required by that version update, see the following section in the JBoss EAP 6 Migration Guide.</td>
</tr>
<tr>
<td></td>
<td>- Web Services Changes</td>
</tr>
<tr>
<td></td>
<td>JBoss EAP 7 introduced JBossWS 5. See the following section in this guide for required updates.</td>
</tr>
<tr>
<td></td>
<td>- Web Services Applications Changes</td>
</tr>
</tbody>
</table>
## APPENDIX A. REFERENCE MATERIAL

### A.1. JACORB SUBSYSTEM MIGRATION OPERATION WARNINGS

The `migrate` operation is not able to process all resources and attributes. The following table lists some of the warnings you might see when you run either the `migrate` or `describe-migration` operation for the `jacorb` subsystem.

**NOTE**

If you see "Could not migrate" or "Can not migrate" entries in the output of the `migrate` operation, this indicates the migration of the server configuration completed successfully but it was not able to automatically migrate all of the elements and attributes. You must follow the suggestions provided by the "migration-warnings" to modify those configurations.

<table>
<thead>
<tr>
<th>Warning Message</th>
<th>What It Means / How to Fix It</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <strong>iiop</strong> migration can be performed when the server is in <strong>admin-only</strong> mode</td>
<td>The <code>migrate</code> operation requires starting the server in <strong>admin-only</strong> mode, which is done by adding parameter <strong>--admin-only</strong> to the server start command: $ EAP_HOME/bin/standalone.sh --admin-only</td>
</tr>
<tr>
<td>Properties X cannot be emulated using OpenJDK ORB and are not supported</td>
<td>Configuration of the specified property is not supported and is not included in the new <strong>iiop-openjdk</strong> subsystem configuration. The behavior exhibited by this property in the previous release of JBoss EAP is not migrated and the administrator must verify that the new <strong>iiop-openjdk</strong> subsystem in JBoss EAP 7 is able to operate correctly without that behavior. Unsupported properties include: <code>cache-poa-names</code>, <code>cache-typecodes</code>, <code>chunk-custom-rmi-valuetypes</code>, <code>client-timeout</code>, <code>comet</code>, <code>indirection-encoding-disable</code>, <code>iona</code>, <code>lax-boolean-encoding</code>, <code>max-managed-buf-size</code>, <code>max-server-connections</code>, <code>max-threads</code>, <code>outbuf-cache-timeout</code>, <code>outbuf-size</code>, <code>queue-max</code>, <code>queue-min</code>, <code>poa-monitoring</code>, <code>print-version</code>, <code>retries</code>, <code>retry-interval</code>, <code>queue-wait</code>, <code>server-timeout</code>, <code>strict-check-on-tc-creation</code>, <code>use-bom</code>, <code>use-imr</code>.</td>
</tr>
</tbody>
</table>
The properties $X$ use expressions. Configuration properties that are used to resolve those expressions should be transformed manually to the new `iiop-openjdk` subsystem format.

Properties that use expressions must be configured manually by the administrator.

For example, the `jacorb` subsystem in JBoss EAP 6 defined a `giop-minor-version` property. The `iiop-openjdk` subsystem in JBoss EAP 7 defines a `giop-version` property. Suppose the `jacorb` subsystem minor version attribute is set to `${iiop-giop-minor-version}` and the system property is configured in the `standalone.conf` file as `-Diiop-giop-minor-version=1`. After the `migrate` operation, the administrator must change the system property value to `1.1` to ensure the new subsystem is configured correctly.

The message contains the explanation.

### A.2. MESSAGING SUBSYSTEM MIGRATION OPERATION WARNINGS

The `migrate` operation is not able to process all resources and attributes. The following table lists some of the warnings you might see when you run either the `migrate` or `describe-migration` operation for the messaging subsystem.

**NOTE**

If you see "Could not migrate" or "Can not migrate" entries in the output of the `migrate` operation, this indicates the migration of the server configuration completed successfully but it was not able to automatically migrate all of the elements and attributes. You must follow the suggestions provided by the "migration-warnings" to modify those configurations.

<table>
<thead>
<tr>
<th>Warning Message</th>
<th>What It Means / How to Fix It</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can not migrate: the new <code>iiop-openjdk</code> subsystem is already defined</td>
<td>The message contains the explanation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Warning Message</th>
<th>What It Means / How to Fix It</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <code>migrate</code> operation can not be performed: the server must be in <code>admin-only</code> mode</td>
<td>The <code>migrate</code> operation requires starting the server in <code>admin-only</code> mode, which is done by adding parameter <code>--admin-only</code> to the server start command: $ EAP_HOME/bin/standalone.sh --admin-only</td>
</tr>
</tbody>
</table>
### Warning Message | What It Means / How to Fix It
--- | ---
Can not migrate attribute **local-bind-address** from resource *X*. Use instead the **socket-binding** attribute to configure this **broadcast-group**. | The message contains the explanation and how to fix it.

Can not migrate attribute **local-bind-port** from resource *X*. Use instead the **socket-binding** attribute to configure this **broadcast-group**. | The message contains the explanation and how to fix it.

Can not migrate attribute **group-address** from resource *X*. Use instead the **socket-binding** attribute to configure this **broadcast-group**. | The message contains the explanation and how to fix it.

Can not migrate attribute **group-port** from resource *X*. Use instead the **socket-binding** attribute to configure this **broadcast-group**. | The **broadcast-group** resource no longer accepts the **local-bind-address**, **local-bind-port**, **group-address**, or **group-port** attributes. It only accepts a **socket-binding** attribute. The warning is notification that resource *X* has an unsupported attribute. You must manually set the **socket-binding** attribute on the resource and ensure it corresponds to a defined **socket-binding** resource.

Classes providing the *X* are discarded during the migration. To use them in the new **messaging-activemq** subsystem, you will have to extend the Artemis-based **Interceptor**. | Messaging interceptors support is significantly different in JBoss EAP 7. Any interceptors configured in the previous version of the subsystem are discarded during migration. See [Migrate Messaging Interceptors](#) for more information.

Can not migrate the HA configuration of *X*. Its **shared-store** and **backup** attributes holds expressions and it is not possible to determine unambiguously how to create the corresponding **ha-policy** for the messaging-activemq’s server. | This means the **hornetq-server X**’s **shared-store** or **backup** attributes contained an expression, such as $\{xxx\}$, and the migration operation was not able to resolve it to a concrete expression. The value is discarded and the **ha-policy** for the **messaging-activemq** must be updated manually.

Can not migrate attribute **local-bind-address** from resource *X*. Use instead the **socket-binding** attribute to configure this **discovery-group**. | The message contains the explanation and how to fix it.

Can not migrate attribute **local-bind-port** from resource *X*. Use instead the **socket-binding** attribute to configure this **discovery-group**. | The message contains the explanation and how to fix it.
<table>
<thead>
<tr>
<th>Warning Message</th>
<th>What It Means / How to Fix It</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can not migrate attribute <em>group-address</em> from resource X. Use instead the <em>socket-binding</em> attribute to configure this <em>discovery-group</em>.</td>
<td>The message contains the explanation and how to fix it.</td>
</tr>
<tr>
<td>Can not migrate attribute <em>group-port</em> from resource X. Use instead the <em>socket-binding</em> attribute to configure this <em>discovery-group</em>.</td>
<td>The <em>discovery-group</em> resources no longer accept <em>local-bind-address</em>, <em>local-bind-port</em>, <em>group-address</em>, or <em>group-port</em> attributes. It only accepts a <em>socket-binding</em>. The warning is notification that resource X has an unsupported attribute. You must manually set the <em>socket-binding</em> attribute on the resource and ensures it corresponds to a defined <em>socket-binding</em> resource.</td>
</tr>
<tr>
<td>Can not create a <em>legacy-connection-factory</em> based on <em>connection-factory</em> X. It uses a HornetQ <em>in-vm</em> connector that is not compatible with Artemis <em>in-vm</em> connector.</td>
<td>The legacy HornetQ remote <em>connection-factory</em> resources are migrated into <em>legacy-connection-factory</em> resources to allow JBoss EAP 6 clients to connect to JBoss EAP 7. However, <em>legacy-connection-factory</em> resources are only created when the <em>connection-factory</em> is using remote connectors. Any <em>connection-factory</em> using <em>in-vm</em> is not migrated because <em>in-vm</em> clients are based on JBoss EAP 7, not JBoss EAP 6. This warning is notification that the <em>in-vm connection-factory</em> was not migrated.</td>
</tr>
</tbody>
</table>
| Can not migrate attribute X from resource Y. The attribute uses an expression that can be resolved differently depending on system properties. After migration, this attribute must be added back with an actual value instead of the expression. | This warning appears when the migration can not resolve attribute X to a concrete value during the migration process. The value is discarded and the attribute must be migrated manually. This happens in the following cases:  
- **cluster-connection forward-when-no-consumers**:  
  This boolean attribute has been replaced by the *message-load-balancing-type* attribute, which is an enum with a value of OFF, STRICT, or ON_DEMAND.  
- **broadcast-group** and **discovery-group’s jgroups-stack** and **jgroups-channel** attributes  
  They reference other resources and JBoss EAP 7 no longer accepts these expressions. |
Can not migrate attribute \( X \) from resource \( Y \). This attribute is not supported by the new \texttt{messaging-activemq} subsystem.

Some attributes are no longer supported in the new \texttt{messaging-activemq} subsystem and are simply discarded:

- hornetq-server's failback-delay
- http-connector's use-nio attribute
- http-acceptor's use-nio attribute
- remote-connector's use-nio attribute
- remote-acceptor's use-nio attribute

Can not migrate attribute \texttt{failback-delay} from resource \( X \). Artemis detects failback deterministically and it no longer requires to specify a delay for failback to occur.

The message contains the explanation.

\begin{table}
\centering
\begin{tabular}{|p{0.4\textwidth}|p{0.5\textwidth}|}
\hline
\textbf{Warning Message} & \textbf{What It Means / How to Fix It} \\
\hline
Can not migrate attribute \( X \) from resource \( Y \). This attribute is not supported by the new \texttt{messaging-activemq} subsystem. & Some attributes are no longer supported in the new \texttt{messaging-activemq} subsystem and are simply discarded: \\
& - hornetq-server's failback-delay \\
& - http-connector's use-nio attribute \\
& - http-acceptor's use-nio attribute \\
& - remote-connector's use-nio attribute \\
& - remote-acceptor's use-nio attribute \\
\hline
Can not migrate attribute \texttt{failback-delay} from resource \( X \). Artemis detects failback deterministically and it no longer requires to specify a delay for failback to occur. & The message contains the explanation. \\
\hline
\end{tabular}
\end{table}

**Replace the Deprecated broadcast-group or discovery-group Attributes**

If you are advised to replace the deprecated \texttt{broadcast-group} or \texttt{discovery-group} attributes with the \texttt{socket-binding} attribute, you can add the new attribute using the management CLI.

This example assumes you are migrating a standalone server that contains the following \texttt{discovery-group} configuration in the \texttt{messaging} subsystem.

\begin{verbatim}
<discovery-groups>
  <discovery-group name="my-discovery-group">
    <group-address>224.0.1.105</group-address>
    <group-port>56789</group-port>
  </discovery-group>
</discovery-groups>
\end{verbatim}

When you run the \texttt{migrate} operation for the \texttt{messaging} subsystem, you see the following output and warnings:

\begin{verbatim}
[standalone@localhost:9999 /] /subsystem=messaging:migrate
{
   "outcome" => "success",
   "result" => {"migration-warnings" => [  
   "WFLYMSG0084: Can not migrate attribute group-address from resource [  
   {"subsystem" => "messaging-activemq"},  
   {"server" => "default"},  
   {"discovery-group" => "my-discovery-group"}  
   ]. Use instead the socket-binding attribute to configure this discovery-group.",  
   "WFLYMSG0084: Can not migrate attribute group-port from resource [  
   {"subsystem" => "messaging-activemq"},  
   {"server" => "default"},
\end{verbatim}
The migrate operation creates a discovery-group named "my-discovery-group" in the new messaging-activemq subsystem that is now configured like the following.

```
<discovery-group name="my-discovery-group"/>
```

You must now use the following management CLI command to create a socket-binding element in the server configuration file named "my-discovery-group-socket-binding".

```
/socket-binding-group=standard-sockets/socket-binding=my-discovery-group-socket-binding:add(multicast-address=224.0.1.105, multicast-port=56789)
```

Next, add the newly created socket-binding to the discovery-group named "my-discovery-group" in the messaging-activemq subsystem in the server configuration file using the following management CLI command.

```
/subsystem=messaging-activemq/server=default/discovery-group=my-discovery-group:write-attribute(name=socket-binding,value=my-discovery-group-socket-binding)
```

These commands create the following XML in the server configuration file.

```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">
  <server name="default">
    ...
    <discovery-group name="my-discovery-group" socket-binding="my-discovery-group-socket-binding"/>
    ...
  </server>
</subsystem>
...
<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">
  ...
  <socket-binding name="my-discovery-group-socket-binding" multicast-address="224.0.1.105" multicast-port="56789"/>
  ...
</socket-binding-group>
```

A.3. WEB SUBSYSTEM MIGRATION OPERATION WARNINGS

The migrate operation is not able to process all resources and attributes. The following table lists some of the warnings you might see when you run either the migrate or describe-migration operation for the web subsystem.
NOTE

If you see "Could not migrate" or "Can not migrate" entries in the output of the `migrate` operation, this indicates the migration of the server configuration completed successfully but it was not able to automatically migrate all of the elements and attributes. You must follow the suggestions provided by the "migration-warnings" to modify those configurations.

<table>
<thead>
<tr>
<th>Warning Message</th>
<th>What It Means / How to Fix It</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate operation only allowed in admin only mode</td>
<td>The <code>migrate</code> operation requires starting the server in <code>admin-only</code> mode, which is done by adding parameter <code>--admin-only</code> to the server start command: $ EAP_HOME/bin/standalone.sh --admin-only</td>
</tr>
<tr>
<td>Could not migrate resource X</td>
<td>The behavior exhibited by this resource in the previous release of JBoss EAP was not migrated. The administrator must verify if the new <code>undertow</code> subsystem in JBoss EAP 7 is able to operate correctly without that behavior or whether the behavior must be migrated manually.</td>
</tr>
<tr>
<td>Could not migrate attribute X from resource Y.</td>
<td>The behavior exhibited by this resource attribute in the previous release of JBoss EAP was not migrated. The administrator must verify if the new <code>undertow</code> subsystem in JBoss EAP 7 is able to operate correctly without that behavior or whether the behavior must be migrated manually. See Web Subsystem Migration Operation Attribute Warnings for the list of attributes that are not migrated.</td>
</tr>
<tr>
<td>Could not migrate SSL connector as no SSL config is defined</td>
<td>The message contains the explanation.</td>
</tr>
<tr>
<td>Could not migrate <code>verify-client</code> attribute X to the Undertow equivalent</td>
<td>The message contains the explanation.</td>
</tr>
<tr>
<td>Could not migrate <code>verify-client</code> expression X</td>
<td>The message contains the explanation.</td>
</tr>
<tr>
<td>Warning Message</td>
<td>What It Means / How to Fix It</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------</td>
</tr>
</tbody>
</table>
| Could not migrate valve X | The behavior exhibited by this valve in the previous release of JBoss EAP was not migrated. The administrator must verify if the new **undertow** subsystem in JBoss EAP 7 is able to operate correctly without that behavior or whether the behavior must be migrated manually. This warning can occur for the following valves:  
  - `org.apache.catalina.valves.RemoteAddrValve`  
    It must have at least one allowed or denied value.  
  - `org.apache.catalina.valves.RemoteHostValve`  
    It must have at least one allowed or denied value.  
  - `org.apache.catalina.authenticator.BasicAuthenticator`  
  - `org.apache.catalina.authenticator.DigestAuthenticator`  
  - `org.apache.catalina.authenticator.FormAuthenticator`  
  - `org.apache.catalina.authenticator.SSLAuthenticator`  
  - `org.apache.catalina.authenticator.SpnegoAuthenticator`  
  - custom valves |
<table>
<thead>
<tr>
<th>Warning Message</th>
<th>What It Means / How to Fix It</th>
</tr>
</thead>
</table>
| Could not migrate attribute X from valve Y | The behavior exhibited by this valve attribute in the previous release of JBoss EAP was not migrated. The administrator must verify if the new *undertow* subsystem in JBoss EAP 7 is able to operate correctly without that behavior or whether the behavior must be migrated manually. This warning can occur for the following valve attributes:  
  - `org.apache.catalina.valves.AccessLogValve`  
    - `resolveHosts`  
    - `fileDateFormat`  
    - `renameOnRotate`  
    - `encoding`  
    - `locale`  
    - `requestAttributesEnabled`  
    - `buffered`  
  - `org.apache.catalina.valves.ExtendedAccessLogValve`  
    - `resolveHosts`  
    - `fileDateFormat`  
    - `renameOnRotate`  
    - `encoding`  
    - `locale`  
    - `requestAttributesEnabled`  
    - `buffered`  
  - `org.apache.catalina.valves.RemoteIpValve`  
    - `httpServerPort`  
    - `httpsServerPort`  
    - `remoteIpHeader`  
      If it is defined but not set to "x-forwarded-for"  
    - `protocolHeader`  
      If it is defined but not set to "x-forwarded-proto" |
Web Subsystem Migration Operation Attribute Warnings

The `migrate` operation is not able to process all JBoss Web attributes. See the following reference tables for information about how to migrate the unprocessed attributes manually.

Web SSL Connector Attributes

The following attributes were used in JBoss EAP 6 to configure the SSL connector. OpenSSL native libraries are not supported in JBoss EAP 7 so there are no equivalent settings.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Undertow Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>ca-revocation-url</td>
<td>The file or URL that contains the revocation list.</td>
<td>No equivalent in Undertow.</td>
</tr>
<tr>
<td>certificate-file</td>
<td>When using OpenSSL encryption, the path to the file containing the server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>certificate.</td>
<td></td>
</tr>
<tr>
<td>ssl-protocol</td>
<td>The SSL protocol string.</td>
<td></td>
</tr>
<tr>
<td>verify-depth</td>
<td>The maximum number of intermediate certificate issuers checked before</td>
<td></td>
</tr>
<tr>
<td></td>
<td>deciding that the clients do not have a valid certificate.</td>
<td></td>
</tr>
</tbody>
</table>

Web Static Resource Attributes

The following `static-resources` element attributes were used to describe how static resources were handled by the DefaultServlet or by the WebdavServlet. There are no equivalents for these attributes because WebDAV is not supported by Undertow. For more information, see https://issues.jboss.org/browse/JBEAP-1036.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Undertow Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>disabled</td>
<td>Enable the default Servlet mapping.</td>
<td>No equivalent setting in Undertow.</td>
</tr>
<tr>
<td>file-encoding</td>
<td>File encoding to be used when reading static files.</td>
<td>No equivalent setting in Undertow.</td>
</tr>
<tr>
<td>max-depth</td>
<td>Maximum recursion for PROPFIND.</td>
<td>This is a WebDAV setting and WebDAV is not supported by Undertow.</td>
</tr>
<tr>
<td>read-only</td>
<td>Allow write HTTP methods (PUT, DELETE).</td>
<td>This is a WebDAV setting and WebDAV is not supported by Undertow.</td>
</tr>
<tr>
<td>secret</td>
<td>Secret for WebDAV locking operations.</td>
<td>This is a WebDAV setting and WebDAV is not supported by Undertow.</td>
</tr>
<tr>
<td>sendfile</td>
<td>Enable sendfile if possible, for files bigger than the specified byte size.</td>
<td>This is set to a sensible default value in Undertow and is not configurable.</td>
</tr>
</tbody>
</table>
Enable WebDAV functionality. | WebDAV is not supported by Undertow.

Web SSO Resource Attributes
SSO is handled differently than in the previous release and there are no equivalent attribute settings in JBoss EAP 7.

<table>
<thead>
<tr>
<th>JBoss Web Attribute</th>
<th>Description</th>
<th>Undertow Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache-container</td>
<td>Name of the cache container to use for clustered SSO.</td>
<td>This setting is no longer needed in Undertow. This works by default across a distributed clustered environment.</td>
</tr>
<tr>
<td>cache-name</td>
<td>Name of the cache to use for clustered SSO.</td>
<td>This setting is no longer needed in Undertow. This works by default across a distributed clustered environment.</td>
</tr>
<tr>
<td>reauthenticate</td>
<td>Whether each request should cause a reauthentication.</td>
<td>There is no equivalent setting in Undertow, which behaves similarly to the <code>reauthenticate=true</code> setting in JBoss EAP 6. While <code>reauthenticate=false</code> could possibly improve performance, it could also create security issues.</td>
</tr>
</tbody>
</table>

Web Access Log Attributes

<table>
<thead>
<tr>
<th>JBoss Web Attribute</th>
<th>Description</th>
<th>Undertow Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>resolve-hosts</td>
<td>Whether to enable resolving hosts for access logging.</td>
<td>Use the setting on the connector to accomplish the same behavior.</td>
</tr>
</tbody>
</table>

Web Connector Attributes

<table>
<thead>
<tr>
<th>JBoss Web Attribute</th>
<th>Description</th>
<th>Undertow Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>executor</td>
<td>The name of the executor that should be used to process the threads of this connector.</td>
<td>You now reference a worker that is defined in the <code>io</code> subsystem. See Migrate the Threads Subsystem Configuration for more information.</td>
</tr>
</tbody>
</table>
proxy-binding
The socket binding to define the host and port that is used when sending a redirect.
There is no direct equivalent.
See https-listener Attributes in the JBoss EAP Configuration Guide for available configuration options.

redirect-port
The port for redirection to a secure connector.
This attribute was deprecated in JBoss EAP 6 and replaced with redirect-binding. Undertow provides the redirect-socket attribute on the http-listener element, which is a replacement for redirect-binding.
See https-listener Attributes in the JBoss EAP Configuration Guide for more information.

A.4. MIGRATE JBOSS WEB SYSTEM PROPERTIES REFERENCE

This reference describes how to map system properties previously used for JBoss Web configuration to the equivalent configuration for Undertow in JBoss EAP 7.

- Map Servlet Container and Connectors System Properties
- Map EL System Properties
- Map JSP System Properties
- Map Security System Properties

Table A.1. Map Servlet Container and Connectors System Properties

<table>
<thead>
<tr>
<th>JBoss EAP 6 System Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jvmRoute</td>
<td>Provides a default value for the jvmRoute attribute. It does not override the automatically generated value when using the standalone-ha.xml configuration file. It supports reload.</td>
</tr>
</tbody>
</table>

Management CLI command:
```
/subsystem=undertow:write-attribute(name=instance-id, value=VALUE)
```
<table>
<thead>
<tr>
<th>Configuration Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.tomcat.util.buf.StringCache.byte.enabled</td>
<td>If <strong>true</strong>, the String cache is enabled for <code>ByteChunk</code>. If the value is not specified, the default value of <strong>false</strong> is used.</td>
</tr>
<tr>
<td>org.apache.tomcat.util.buf.StringCache.char.enabled</td>
<td>If <strong>true</strong>, the String cache is enabled for <code>CharChunk</code>. If the value is not specified, the default value of <strong>false</strong> is used.</td>
</tr>
<tr>
<td>org.apache.tomcat.util.buf.StringCache.cacheSize</td>
<td>The size of the String cache. If the value is not specified, the default value of <strong>5000</strong> is used.</td>
</tr>
<tr>
<td>org.apache.tomcat.util.buf.StringCache.maxStringLength</td>
<td>The maximum length of String that will be cached. If the value is not specified, the default value of <strong>128</strong> is used.</td>
</tr>
<tr>
<td>org.apache.tomcat.util.http.FastHttpDateFormat.CACHE_SIZE</td>
<td>The size of the cache to use parsed and formatted date value. If the value is not specified, the default value of <strong>1000</strong> is used.</td>
</tr>
<tr>
<td>org.apache.catalina.core.StandardService.DELAY_CONNECTOR_STARTUP</td>
<td>If <strong>true</strong>, the connector startup is not done automatically. It is useful in embedded mode.</td>
</tr>
<tr>
<td>org.apache.catalina.connector.Request.SESSION_ID_CHECK</td>
<td>If <strong>true</strong>, the Servlet container verifies that a session exists in a context with the specified session ID before creating a session with that ID.</td>
</tr>
<tr>
<td>org.apache.coyote.USE_CUSTOM_STATUS_MSG_IN_HEADER</td>
<td>If <strong>true</strong>, custom HTTP status messages are used within HTTP headers. Users must ensure that any such message is <strong>ISO-8859-1</strong> encoded, particularly if user provided input is included in the message, to prevent a possible XSS vulnerability. If value is not specified the default value of <strong>false</strong> is used.</td>
</tr>
</tbody>
</table>

Must be enabled programmatically by implementing a custom `io.undertow.servlet.ServletExtension`. Then use the extension to call `setSendCustomReasonPhraseOnError(true)` on the `io.undertow.servlet.api.DeploymentInfo` structure instance.
<table>
<thead>
<tr>
<th>Class/Field</th>
<th>Description</th>
<th>Management CLI command</th>
</tr>
</thead>
</table>
| org.apache.tomcat.util.http.Parameters.MAX_COUNT | The maximum number of parameters that can be parsed in a post body. If exceeded, parsing fails using an `IllegalArgumentException`. The default value is 512 parameters. | ```/subsystem=undertow/server=default-server/http-listener=default:write-attribute(name=max-parameters,value=VALUE)
/subsystem=undertow/server=default-server/https-listener=default:write-attribute(name=max-parameters,value=VALUE)
/subsystem=undertow/server=default-server/ajp-listener=default:write-attribute(name=max-parameters,value=VALUE)``` |
| org.apache.tomcat.util.http.MimeHeaders.MAX_COUNT | The maximum number of headers that can be sent in the HTTP request. If exceeded, parsing will fail using an `IllegalArgumentException`. The default value is 128 headers. | ```/subsystem=undertow/server=default-server/http-listener=default:write-attribute(name=max-headers,value=VALUE)
/subsystem=undertow/server=default-server/https-listener=default:write-attribute(name=max-headers,value=VALUE)
/subsystem=undertow/server=default-server/ajp-listener=default:write-attribute(name=max-headers,value=VALUE)``` |
| org.apache.tomcat.util.net.MAX_THREADS          | The maximum number of threads a connector is going to use to process requests. The default value is \(32 \times 512\). \((512 \times \text{Runtime.getRuntime().availableProcessors()})\) for the JIO connector. | ```/subsystem=io/worker=default:write-attribute(name=task-max-threads,value=VALUE)``` |
| org.apache.coyote.http11.Http11Protocol.MAX_HEADER_SIZE | The maximum size of the HTTP headers, in bytes. If exceeded, parsing will fail using an `ArrayOutOfBoundsException`. The default value is 8192 bytes. | ```/``` |

Red Hat JBoss Enterprise Application Platform 7.0 Migration Guide
<table>
<thead>
<tr>
<th>Management CLI command:</th>
</tr>
</thead>
<tbody>
<tr>
<td>/subsystem=undertow/server=default-server/http-listener=default:write-attribute(name=max-header-size,value=VALUE)</td>
</tr>
<tr>
<td>/subsystem=undertow/server=default-server/https-listener=default:write-attribute(name=max-header-size,value=VALUE)</td>
</tr>
<tr>
<td>/subsystem=undertow/server=default-server/ajp-listener=default:write-attribute(name=max-header-size,value=VALUE)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Allows using simple compression with the HTTP connector. The default value is off, and compression can be enabled using the value on to enable it conditionally, or force to always enable it.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Configure a filter using the management CLI:</th>
</tr>
</thead>
<tbody>
<tr>
<td># Create a filter</td>
</tr>
<tr>
<td>/subsystem=undertow/configuration=filter/gzip=gzipfilter:add()</td>
</tr>
<tr>
<td>/subsystem=undertow/server=default-server/host=default-host/filter-ref=gzipfilter:add()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User agents regexps that will not receive compressed content. The default value is empty.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Configure a predicate in a filter using the management CLI:</th>
</tr>
</thead>
<tbody>
<tr>
<td># Use a predicate in a filter</td>
</tr>
<tr>
<td>/subsystem=undertow/configuration=filter/gzip=gzipfilter:add()</td>
</tr>
<tr>
<td>/subsystem=undertow/server=default-server/host=default-host/filter-ref=gzipfilter:add(predicate=&quot;regex[patte%{i,User-Agent}]&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Content type prefixes of compressible content. The default value is text/html, text/xml, text/plain.</td>
</tr>
</tbody>
</table>
Configure a predicate in a filter using the management CLI:

```java
# Use a predicate in a filter
/subsystem=undertow/configuration=filter/
gzip=gzipfilter:add()
/subsystem=undertow/server=default-
server/host=default-host/filter-
ref=gzipfilter:add(predicate="regex[patte
rn='text/html',value=%{o,Content-Type}]")
```

Minimum size of content that will be compressed. The default value is **2048** bytes.

**org.apache.coyote.http11.DEFAULT_CONNECTION_TIMEOUT**  
Default socket timeout. The default value is **60000** ms.

**Management CLI command:**

```java
/subsystem=undertow/server=default-
server/http-listener=default:write-
attribute(name=no-request-
timeout,value=VALUE)
/subsystem=undertow/server=default-
server/https-listener=default:write-
attribute(name=no-request-
timeout,value=VALUE)
/subsystem=undertow/server=default-
server/ajp-listener=default:write-
attribute(name=no-request-
timeout,value=VALUE)
```

**org.jboss.as.web.deployment.DELETE_WORK_DIR_ONCONTEXTDESTROY**  
Use this property to remove .java and .class files to ensure that JSP sources are recompiled. The default value is **false**. Default socket timeout for keep-alive. The default value is **-1** ms, which means it will use the default socket timeout.

No equivalent configuration
<table>
<thead>
<tr>
<th>JBoss EAP 6 System Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.apache.tomcat.util.buf.StringCache.trainThreshold</code></td>
<td>Specifies the number of times <code>toString()</code> must be invoked before activating cache. The default value is <code>100000</code>.</td>
</tr>
<tr>
<td></td>
<td>No equivalent configuration</td>
</tr>
</tbody>
</table>

**Table A.2. Map EL System Properties**

<table>
<thead>
<tr>
<th>JBoss EAP 6 System Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.apache.el.parser.COERCE_TO_ZERO</code></td>
<td>If <code>true</code>, when coercing expressions to numbers, empty strings (&quot;&quot;) and null will be coerced to zero as required by the specification. If a value is not specified, the default value of <code>true</code> is used.</td>
</tr>
<tr>
<td></td>
<td>System property is still valid and processed by the EL</td>
</tr>
</tbody>
</table>

**Table A.3. Map JSP System Properties**

<table>
<thead>
<tr>
<th>JBoss EAP 6 System Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.apache.jasper.compiler.Generator.VARIABLE_EXPRESSIONFACTORY</code></td>
<td>The name of the variable to use for the expression language expression factory. If value is not specified, the default value of <code>_el_expressionfactory</code> is used.</td>
</tr>
<tr>
<td></td>
<td>System property has not changed</td>
</tr>
<tr>
<td><code>org.apache.jasper.compiler.Generator.VARIABLE_INSTANCEMANAGER</code></td>
<td>The name of the variable to use for the instance manager factory. If value is not specified, the default value of <code>_jspinstancemanager</code> is used.</td>
</tr>
<tr>
<td></td>
<td>System property has not changed</td>
</tr>
<tr>
<td><code>org.apache.jasper.compiler.Parser.STRICT_QUOTE_ESCAPING</code></td>
<td>If <code>false</code>, the requirements for escaping quotes in JSP attributes are relaxed so that a missing required quote does not cause an error. If value is not specified, the specification compliant default of <code>true</code> is used.</td>
</tr>
<tr>
<td></td>
<td>System property has not changed</td>
</tr>
<tr>
<td><code>org.apache.jasper.Constants.DEFAULT_TAG_BUFFER_SIZE</code></td>
<td>Any tag buffer that expands beyond <code>org.apache.jasper.Constants.DEFAULT_TAG_BUFFER_SIZE</code> is destroyed and a new buffer is created of the default size. If value is not specified, the default value of <code>512</code> is used.</td>
</tr>
<tr>
<td>System property has not changed</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>org.apache.jasper.runtime.JspFactoryIm pl.USE_POOL</strong></td>
<td>If true, a ThreadLocal PageContext pool is used. If value is not specified, the default value of true is used.</td>
</tr>
<tr>
<td>System property has not changed</td>
<td></td>
</tr>
<tr>
<td><strong>org.apache.jasper.runtime.JspFactoryIm pl.POOL_SIZE</strong></td>
<td>The size of the ThreadLocal PageContext. If value is not specified, the default value of 8 is used.</td>
</tr>
<tr>
<td>System property has not changed</td>
<td></td>
</tr>
<tr>
<td><strong>org.apache.jasper.Constants.JSP_SERVLET_BASE</strong></td>
<td>The base class of the Servlets generated from the JSPs. If value is not specified, the default value of org.apache.jasper.runtime.HttpJspBase is used.</td>
</tr>
<tr>
<td>System property has not changed</td>
<td></td>
</tr>
<tr>
<td><strong>org.apache.jasper.Constants.SERVICE_METHOD_NAME</strong></td>
<td>The name of the service method called by the base class. If value is not specified, the default value of _jspService is used.</td>
</tr>
<tr>
<td>System property has not changed</td>
<td></td>
</tr>
<tr>
<td><strong>org.apache.jasper.Constants.SERVLET_CLASSPATH</strong></td>
<td>The name of the ServletContext attribute that provides the class path for the JSP. If value is not specified, the default value of org.apache.catalina.jsp_classpath is used.</td>
</tr>
<tr>
<td>System property has not changed</td>
<td></td>
</tr>
<tr>
<td><strong>org.apache.jasper.Constants.JSP_FILE</strong></td>
<td>The name of the request attribute for &lt;jsp-file&gt; element of a servlet definition. If present on a request, this overrides the value returned by request.getServletPath() to select the JSP page to be executed. If value is not specified, the default value of org.apache.catalina.jsp_file is used.</td>
</tr>
<tr>
<td>System property has not changed</td>
<td></td>
</tr>
<tr>
<td><strong>org.apache.jasper.Constants.PRECOMPIL E</strong></td>
<td>The name of the query parameter that causes the JSP engine to just pregenerate the servlet but not invoke it. If value is not specified, the default value of org.apache.catalina.jsp_precompile is used.</td>
</tr>
<tr>
<td>System property has not changed</td>
<td></td>
</tr>
<tr>
<td><strong>org.apache.jasper.Constants.JSP_PACKAGE_NAME</strong></td>
<td>The default package name for compiled JSP pages. If value not specified, the default value of org.apache.jsp is used.</td>
</tr>
<tr>
<td>JBoss EAP 6 System Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>org.apache.catalina.connector.RECYCLE_FACADES</td>
<td>If this is \texttt{true} or if a security manager is in use a new facade object is created for each request. If value is not specified, the default value of \texttt{false} is used.</td>
</tr>
<tr>
<td>org.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH</td>
<td>If this is \texttt{true} the \texttt{\textbackslash} character is permitted as a path delimiter. If value is not specified, the default value of \texttt{false} is used.</td>
</tr>
<tr>
<td>org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH</td>
<td>If this is \texttt{true}, \texttt{%2F} and \texttt{%5C} is permitted as path delimiters. If value is not specified, the default value of \texttt{false} is used.</td>
</tr>
</tbody>
</table>
Management CLI command:

```
/subsystem=undertow/server=default-server/http-listener=default:write-attribute(name=allow-encoded-slash,value=VALUE)
/subsystem=undertow/server=default-server/https-listener=default:write-attribute(name=allow-encoded-slash,value=VALUE)
/subsystem=undertow/server=default-server/ajp-listener=default:write-attribute(name=allow-encoded-slash,value=VALUE)
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.catalina.STRICT_SERVLET_COMPLIANCE</td>
<td>If value is not specified, true is used. If this is true the following actions will occur: any wrapped request or response object passed to an application dispatcher is checked to ensure that it has wrapped the original request or response. (SRV.8.2 / SRV.14.2.5.1) a call to Response.getWriter() if no character encoding has been specified results in subsequent calls to Response.getCharacterEncoding() returning ISO-8859-1 and the Content-Type response header will include a charset=ISO-8859-1 component. (SRV.15.2.22.1) every request that is associated with a session causes the session's last accessed time to be updated regardless of whether or not the request explicitly accesses the session. (SRV.7.6) Compliant by default</td>
</tr>
<tr>
<td>org.apache.catalina.core.StandardWrapperValve.SERVLET_STATS</td>
<td>If true or if org.apache.catalina.STRICT_SERVLET_COMPLIANCE is true, the wrapper will collect the JSR-77 statistics for individual servlets. If value is not specified, the default value of false is used. No equivalent configuration</td>
</tr>
<tr>
<td>org.apache.catalina.session.StandardSession.ACTIVITY_CHECK</td>
<td>If this is true or if org.apache.catalina.STRICT_SERVLET_COMPLIANCE is true Tomcat tracks the number of active requests for each session. When determining if a session is valid, any session with at least one active request is always be considered valid. If value is not specified, the default value of false is used. No equivalent configuration</td>
</tr>
</tbody>
</table>
A.5. COMPATIBILITY AND INTEROPERABILITY BETWEEN RELEASES

This section describes the compatibility and interoperability of client and server EJB and messaging components between the JBoss EAP 5, JBoss EAP 6, and JBoss EAP 7 releases.

EJB remoting over IIOP
You should not encounter problems with any of the following configurations.

- Connecting from a JBoss EAP 5 client to a JBoss EAP 7 server
- Connecting from a JBoss EAP 6 client to a JBoss EAP 7 server
- Connecting from a JBoss EAP 7 client to a JBoss EAP 6 server
- Connecting from a JBoss EAP 7 client to a JBoss EAP 5 server

EJB remoting Using JNDI
You should not encounter problems with any of the following configurations.

- Connecting from a JBoss EAP 6 client to a JBoss EAP 7 server
- Connecting from a JBoss EAP 7 client to a JBoss EAP 6 server

JBoss EAP 6 provided support for the EJB 3.1 specification and introduced the use of standardized global JNDI namespaces, which are still used in JBoss EAP 7. Due to the change in JNDI namespace names, the following configurations are not compatible:

- Connecting from a JBoss EAP 5 client to a JBoss EAP 7 or a JBoss EAP 6 server
- Connecting from a JBoss EAP 7 or JBoss EAP 6 client to a JBoss EAP 5 server

For more information about standardized JNDI namespace changes, see JNDI Changes in the JBoss EAP 6 Migration Guide.

EJB remoting Using @WebService
You should not encounter problems with any of the following configurations.

- Connecting from a JBoss EAP 5 client to a JBoss EAP 7 server
- Connecting from a JBoss EAP 6 client to a JBoss EAP 7 server
- Connecting from a JBoss EAP 7 client to a JBoss EAP 6 server
- Connecting from a JBoss EAP 7 client to a JBoss EAP 5 server

Messaging Standalone Client
You should not encounter problems with any of the following configurations.

- Connecting from a JBoss EAP 6 client to a JBoss EAP 7 server
- Connecting from a JBoss EAP 7 client to a JBoss EAP 6 server

In the following configuration, if the client is using the messaging broker-specific HornetQ API rather than the generic JMS API, the connection is possible. However, JNDI lookups must be addressed using the JBoss EAP legacy JNDI naming extension that is delivered with JBoss EAP 7.

- Connecting from a JBoss EAP 5 client to a JBoss EAP 7 server
JBoss EAP 7 built-in messaging is not able to connect to HornetQ 2.2.x that shipped with JBoss EAP 5 due to protocol compatibility issues. For this reason, the following configurations are not compatible.

- Connecting from a JBoss EAP 7 client to a JBoss EAP 5 server

**Messaging MDBs**
You should not encounter problems with any of the following configurations.

- Connecting from a JBoss EAP 6 client to a JBoss EAP 7 server
- Connecting from a JBoss EAP 7 client to a JBoss EAP 6 server

In the following configuration, if the client is using the messaging broker-specific HornetQ API rather than the generic JMS API, the connection is possible. However, JNDI lookups must be addressed using the JBoss EAP legacy JNDI naming extension that is delivered with JBoss EAP 7.

- Connecting from a JBoss EAP 5 client to a JBoss EAP 7 server

JBoss EAP 7 built-in messaging is not able to connect to HornetQ 2.2.x that shipped with JBoss EAP 5 due to protocol compatibility issues. For this reason, the following configurations are not compatible.

- Connecting from a JBoss EAP 7 client to a JBoss EAP 5 server

**JMS bridges**
You should not encounter problems with any of the following configurations.

- Connecting from a JBoss EAP 5 client to a JBoss EAP 7 server
- Connecting from a JBoss EAP 6 client to a JBoss EAP 7 server
- Connecting from a JBoss EAP 7 client to a JBoss EAP 6 server
- Connecting from a JBoss EAP 7 client to a JBoss EAP 5 server

*Revised on 2018-02-08 10:18:33 EST*